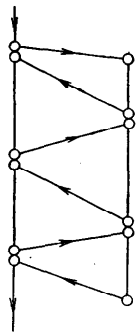
電子計算機技術の動向*

高　橋　　茂**

1. はじめに

「電子計算機技術の動向」という大ざっぱな題をつけたが，ここでは最近の顕著な傾向の一つである時分割（time sharing）とそれに関連する技術に問題を限定したい．

外国の新らしい計算機，Honeywll 800, RCA-601, IBM-7080 などの広告を見ると，必ずプログラムの多重処理（parallel processing of independent programs）ということがうたってある．これは，Univac Larc[1,2] や第6節で触れる NBS の Pillot[3,4] ように，二つ以上の計算機を結合したものではない．プログラムが二つの場合を例にとると，**第1図**に示すように動作するもので，あくまで一つの計算機が二つのプログラムの間を往復しながら，それらを実行しているに過ぎない．ちょっと考えると，これはどちらか一方のプログラムを済ませてから，他方を実行するのと，何の変りもないようにみえるがプログラムに動作の遅い入出力装置が含まれていると事情は全く変って来る．入出力装置の動作が完了するのを待っている間に，他方のプログラムを実行できるからである．



第1図
二つのプログラムでの時分割

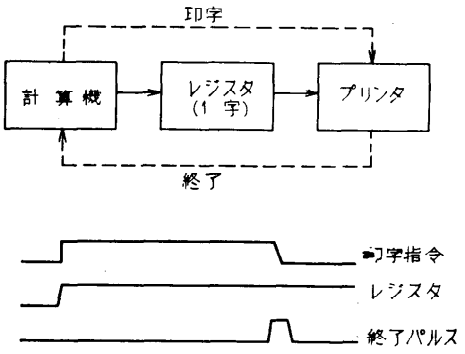
時分割技術は，このように入出力装置が遅いことによる計算機時間の無駄を少なくするために生れて来たものであるが，この考え方を押進めることによって，ディジタル計算機は今までよりも一層融通性に富んだものとなり，しかも金物は節約できることとなった．一方では，このように入出力装置の速度と計算機本体との不平衡を調和させようとして考え出された時分割技術が，逆に計算機本体の一層の高速化を要求するという，一見奇妙な結果も生れて来た．

* Some Technical Trends in Electronic Digital Computers, by Shigeru Takahashi (Electrotechnical Laboratory, Tokyo). 昭和35年10月14日 情報処理学会関西講演会における講演．

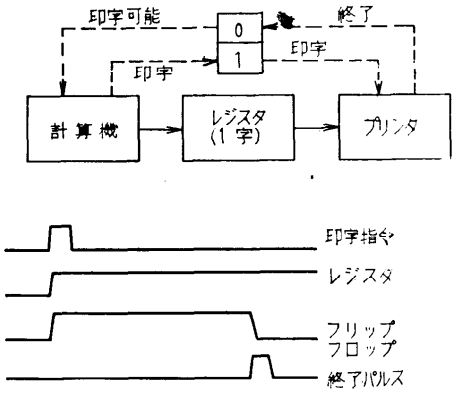
** 電気試験所

2. 時分割方式の例

出力装置として，毎分10字程度の速度の普通のプリンタがついている計算機を考えよう．一つの命令の実行が**完全に**終ってから，次の命令に進むという考え方に立てば，プリンタと計算機との接続は**第2図**のようになる．すなわち，計算機から印字指令パルスを出



第2図　最も簡単な計算機とプリンタとの接続



第3図　おいてけぼり制御

すと同時に1字分のレジスタに情報を送り込む．プリンタはその1字を印字し終ると終了パルスを計算機に送り，計算機はこれを受取ることによって次の命令に進む（レジスタの内容は次の印字の際に入れ変えればよく，特に消す必要はない）．これで計算機は1字印

字するごとにまるまる 100 ms 待たされ，その間何もすることができない．

そこで直ちに考えられるのが，**第3図**のように計算機とプリンタとの間の指令と報告のやりとりを一つのフリップ・フロップを介して行う方法である．この場合，計算機の出力命令はこのフリップ・フロップがオフのときにだけ実行可能であり，もしプログラムがそこまで行ったときに，フリップ・フロップがオンであれば，それがプリンタからの印字終了の報告によってオフになるのをまって，その実行を開始する．その実行はフリップ・フロップをオンにすると同時に，レジスタに1字分の情報を送り込めば終りで，計算機は直ちに次の命令に進むことができる．一方，プリンタはフリップ・フロップに蓄えられた指令によって，自主的（autonomous）にレジスタの内容を印字し，それが終るとフリップ・フロップをオフにする．すなわち計算機はプリンタを“おいてけぼり”にしておくのである．

この方法ではプリンタが印字中に，計算機は他のこと，例えば次の字を打出すための計算をすることができる．この計算が丁度 100 ms ですめば，話は非常にうまいわけで，計算機もプリンタもともに休みなく動くことになる．しかし一般にはそううまくは行かず，計算機が待たされるか，プリンタが待たされるかのどちらかになってしまう．ここで東大の PC-1[5] での例を挙げよう．

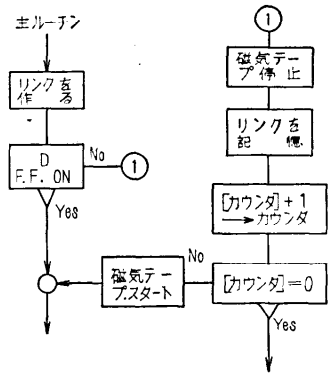
PC-1 では早く印字命令が来すぎたときに，無駄になる計算機時間を有効に使おうとして，出力命令に，“累算器の内容を1字印字せよ．もしプリンタが使用中であれば（すなわち第3図のフリップ・フロップがオンであれば）n 番地に飛越せ”という機能を持たせた．これはプリンタが空いていなければ飛越して，別の計算をやり，**頃合いを見計らって**帰ってこようという趣旨のものであるが，実際には n としてその命令自身を記憶してある場所を書いて，いわゆるダイナミック・ストップの状態で計算機を待たせる人ばかりで，有効には使用されなかった由である．

これは“頃合いを見計らう”ということが，それほど容易ではなく，短かすぎれば再び計算機が待たされ，長すぎれば，たださえ遅いプリンタを待たせることになるからである．しかしプログラミングの面倒さえいとわなければ原理的にはこの方法でプログラムの多重処理ができるわけである．

“割込み”（automatic interruption）の技術を用いると，これらのプログラミング上の面倒を除去すると共に，時分割を一層有効にすることができる．すなわち第3図のフリップ・フロップがオフになるや否や，現在実行中のプログラムに割込む（オフになった瞬間に実行中，あるいは，これから実行されようとする命令が終るのを待って，印字の命令あるいはルーチンに飛越させる）．しかし前述のようなプログラム上の技術だけによる時分割とは異り，割込みを使う場合には，これが何時起るかということはプログラマの与り知らないところであって，その実施については，いくつかの注意事項がある．これらについては第4節に述べる．

時分割技術を最初に使用したのは，恐らくケンブリッジ大学のグループであろう．磁気テープの使用法にはいろいろあるが，イギリスでの使用法は任意にブロック番号を与えたときにそのブロック上でテープを止めること（positioning）を要求している．世界最初のプログラム記憶式計算機として有名な EDSAC-1 には，この positioning を時分割でやる D と呼ばれる命令があった[6]．この命令は前述の PC-1 の命令とよく似ていて，この命令でオンになり，磁気テープからのブロック・マークでオフになるフリップ・フロップに関して判断を行なう．すなわち“このフリップ・フロップがオンならそのまま次に進み，オフなら n 番地に飛越せ”という命令である．

この D 命令を主プログラム中に適当にばらまいておいて，positioning のルーチンに飛越し，また主ルーチンに帰るということを繰り返せば，計算機時間の極く一部分を使うだけで，ほぼ自主的に positioning ができる．**第4図**はその流れ図でカウンタには position-



第4図 EDSAC-1 の磁気テープ positioning の流れ図

ing を開始したときのブロック番号と所望のブロック番号との差を負の値として書込んである（最初にいずれが大きいかによって，テープの前進，後退を決めておく）．

EDSAC-2 ではこれをもっと効果的にやるために，割込みを使っているが，また同時に第4図に示したことを全て金物でやってしまうようになっている．すなわち，ブロック・マークが来るごとに，現在実行中の命令が終了するのを待って，磁気テープを止め，記憶装置の特定の場所を調べて，その内容が −1 ならそのまま，−1 でなければそれに1を加え，テープを再び動かす．これだけのことをプログラムでなく金物でやることは無駄であるようにもみえるが，EDSAC-2 はマイクロ・プログラミング[7]を使った機械であるから，実際には大して金物を増さなくても，このようなことができたのであろう．

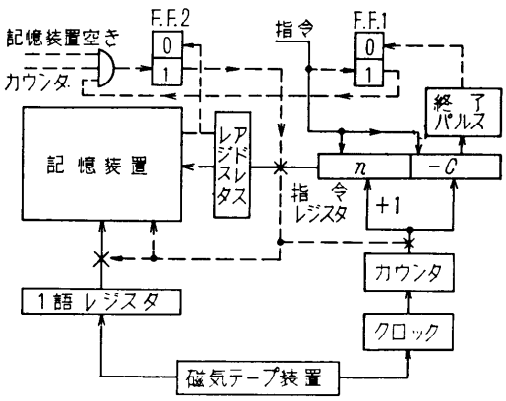
3. 時分割の種々のレベル

今までの説明では，複数個のプログラムが一つの計算機を共用することだけが時分割であるかのような印象を与えたかもしれない．実際には計算機の速度と入出力装置の速度との兼ね合いで，種々の時分割方式が用いられる．これを磁気テープから1ブロックの情報を読込む場合について説明しよう．

比較のために時分割でないものから述べる．サブルーチンによるにしろ，金物によるにしろ，とにかく1ブロック読むまでは計算機はそれに専念するというのが一つの方法，数十語を1ブロックとし，1ブロックを収容し得るバッファ記憶装置を設け，“読込め”という命令で磁気テープの内容を一たんこれに移し（この動作はほぼ自主的に行われる），別の命令でバッファ記憶装置から主記憶装置に移すのが他の方法である．前者は計算機時間が無駄になり，後者はバッファ記憶装置に金がかかる．

時間も金も節約できるのが時分割方式であるが，それにはまず記憶装置がかなり高速のものでなければならず，現状では磁心マトリクス以外にはない．時分割方式のうち，プログラムには関係がなく，金物としてはまだ最も複雑なものが，記憶装置だけを時分割で使うものである．**第5図**にその概要を示す．

“*C* 語を主記憶装置の *n* 番地以後の場所に読込め”という命令は，フリップ・フロップがオフであれば直ちに実行することができる（オンであれば前の読込みを実施中であるから，済むまで待つ必要がある）．フ



第5図　主記憶装置だけの時分割使用

リップ・フロップ1をオンにし，指令レジスタに *n* と −*C* を入れれば，命令としての実行は終りで，あとは第5図の装置が自主的に読込みを遂行する．磁気テープから1語のレジスタに情報を詰め込み，詰め終ればクロック・パルスを数えているカウンタから出力がでる．主記憶装置は計算機に使用されているが，その duty ratio はそう大きいものではなく，磁気テープの次のクロック・パルスが来るまでには必ず空く．記憶装置が空くとフリップ・フロップ2がオンになり，*n* がアドレス・レジスタに移され，*n* および −*C* にカウンタの出力1が加えられる．

一方，1語のレジスタの内容がアドレス・レジスタによって指定される *n* 番地に書込まれ，フリップ・フロップ2はリセットされる．このフリップ・フロップがオンである間は計算機は記憶装置を使うことができないが，これは時間の割合にすれば極めて小さい（磁気テープの速度を 10 kc，1語の桁数を10字，磁心マトリクスのサイクル時間を 6 μs とすると 0.6%）．以下同様にして磁気テープの情報が次々に記憶装置に読み込まれ，最後に −*C* が0になると，終了パルスが出て，フリップ・フロップ1はリセットされ読込みは終了する．

この方式では記憶装置の時分割が行われ，高価なバッファ記憶装置は不要になっているが，プログラムのレベルでの時分割は行われていない．もちろん，第5図の装置が自主的に動作している間に，計算機はルーチンの実行を継続できるわけであるが，このように専用の金物をぜい沢に設けた場合には，これをプログラムのレベルで時分割であるとはいい得ない．極論すればバッファ記憶装置を設けた場合や，計算機が二つあ

る場合と同じだからである．

第5図の金物のうち，指令レジスタ，終了パルス検出回路およびフリップ・フロップ2を取除き，カウンタの出力でフリップ・フロップ1をリセットして，主プログラムに割込み，磁気テープ読込みサブルーチンに飛越して，アドレス計算，計数，判断などを全てプログラムで実行することができる．こうなればもちろんプログラムのレベルで時分割といえる．

また，さらに高いレベルで考えると，レジスタは1字分でよく，カウンタも不要で，磁気テープからのクロック・パルスで割込み，語の編集もプログラムに委ねることとなろう[8]．

いずれのレベルを採用すべきかは，計算機の速度と磁気テープの速度との比により，また金物の複雑さとプログラムの複雑さとの得失による．これらについては第7節に述べる．

市販の計算機では，例えば IBM-709/7090 では，data synchronizer というものが第5図の回路の役割をしており，また IBM-7070 では，これにさらに data scattering の機能がつき，ますます金物が増えている．電気試験所で試作中の ETL Mk-4 B[9] でもテープ制御装置と称する同様のものがあり，いずれも磁気テープ関係の時分割については最下位のレベルに属する．

MIT の TX-2[10]は，時分割並びに割込みの技術を全面的に採用した最初の計算機であるが，磁気テープを扱う場合の時分割のレベルは最上位，すなわち字ごとにプログラムによって受取るものであるといわれている．

4. 割込みの諸方式

割込みがいつ起るかということは，プログラマには予知できない（できるくらいなら割込みは不要）からその実施については次のような注意事項がある．

（1） あとでもとのルーチンに戻れるように，どこから飛越したかというアドレスを記憶する．

（2） 割込みを生ずる状態が複数個あるときの優先権を指定する．

（3） 一度割込みが起ると，準備ができるまで次の割込みを禁止する．

（4） 飛越した先で累算器その他の演算レジスタを使用するときには，その内容を保存する．

これらの事項をどのように実施するか，またどのようにして（主として金物によるか，プログラムによるか）実施するかによって，種々の割込方式が生れてくる．しかし(4)については，現在用いられているどの方式でも，全てプログラムにより，必要に応じてこれらを保存しているようである．以下(1)，(2)，(3)について考察する．

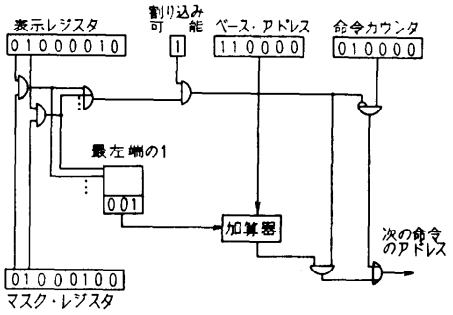
前述の TX-2 は主として金物により，これらを次のように実施している．

（1） 32のルーチンがあり，それぞれに命令カウンタがついている．割込みがあると命令カウンタを取替えるだけで，元のルーチンのどのステップにあったかは，それに属する命令カウンタによって記憶されている．

（2） 各ルーチンには固有の優先順序がついている．あるルーチンには特定の入出力装置が含まれるが，それらの間の優先順序は，含まれている入出力装置の起動停止の困難さの順になっている．すなわち磁気テープ装置などは最も優先順位が高い．

（3） 各命令にその命令終了後高い優先順位のルーチンに移っても差支えないことを示すビット（break bit）あるいは仕事が済んで低い優先順位の仕事に移ってもよいことを示すビット（dismiss bit）がついている．

IBM の Stretch[11] も割込み技術を全面的に採用している計算機であるが，その行き方は TX-2 とは逆に，できるだけプログラムによっている．Stretch には64ビットの表示レジスタと，同じく64ビットのマスク・レジスタというものがある．これらには共にアドレスがつけられ，データとして扱える．表示レジスタの各ビットは，それに対応する入出力装置その他の割込みを必要とする状態の出現によって1になり，計算機がその割込みを許せば0になる．割込みが許されるか許されないかは，マスク・レジスタの対応するビットに1があるか0があるかによる．第6図に示すよ



第6図　IBM stretch の割り込み方式

うに，一つの命令終了後（命令カウンタの内容は割り込みがないときの次の命令の所在を示している），表示レジスタの内容とマスク・レジスタの内容とを較べ共に1のビットをもつ桁があれば，その最左端のものを求め，左端から数えたその桁数をベース・アドレス（プログラムによって与える）に加えたものを，次の命令のアドレスとする．

（1） 上述のように，割込み直後の命令の所在は，第6図の加算器の出力によって直接指定するから，命令カウンタの内容はまだ保存されている．したがって加算器の出力によって指定された命令を，ただ一つだけ実行してすぐ元に戻ることもできるし，“命令カウンタの内容を n 番地に記憶して，m 番地に飛越せ”という命令で本当に飛越すこともできる（戻るべきアドレスは n 番地に記憶される）．

（2） 表示レジスタの左から優先順位がついている．

（3） マスク・レジスタによって制御できるほか，別に割込み可能フリップ・フロップというものがあって，プログラムによってオン・オフでき，これがオフであれば，割込みはいかなる場合でも禁止される．

IBM-7070 は入出力の割込み機能を設けた市販の計算機としては最初のものと思われるが，これはStretch などに較べると，かなり簡易化されている．

（1） 主ルーチンにだけ割込みが許される．主ルーチンの次の命令の所在は，特定の指標レジスタに蓄えられる．

（2） 一定の順序と速度で，次々に表示灯を調べる回路があり，これに引かかったものから割込みが許される．したがって特に優先権はない．マスクは表示灯のグループにつけられる．

（3） 一たん優先ルーチンに入ると，他のものの割込みは一切禁止される．

前述の PC-1 には，その後割込み機能がつけられた[12]．割込むのはプリンタだけであるから，優先権の問題はない．主ルーチンの次の命令の所在は，記憶装置の特定の場所に記憶され，一たん割込みが起ると，プログラムで解除するまで，次の割込みは禁止される．

ETL Mk-4 B[9]では Stretch とほぼ同様の方法を採用しているが，次の点がちがう．すなわち割込みが起ると表示レジスタの最右端のビットに対応する記憶場所の左半分に今までの命令カウンタの内容を蓄え，同じ場所の右半分の内容をとり出して，命令カウンタにセットする．これで帰りのアドレスが保存され，かつ飛越しが起ったことになる．

市販の国産機では NEAC-2203 に割込み機能がついていて，時分割により3種のプログラムを扱うことができる[13]．

5. 割込み技術の拡張[14]

割込みは入出力装置についての時分割の効率を高めるだけではない．累算器の“あふれ”，回路の誤動作その他の条件で割込んで，これらを修正したり，原因を調べたりするルーチンに飛ぶことはすでに行われている．他にも次のような応用が考えられる．

（1） プログラム検査への応用．プログラムを機械にかけて，手動で命令を1ステップずつ実行してみるという手段は，プログラマやコーダにとっては都合のよいものであるが，従来は計算時間を最も無駄にするものとして，常に排撃されて来た．割込み技術によって計算機時間を殆んど無駄にすることなく，何本ものプログラムを上のようにして検査できる．またあるスイッチをセットすると，命令を一つ実行するごとに，あるいは特色の命令，たとえば飛越命令を実行するごとに，割込むようにして，スイッチの数だけの種々の検査プログラムで，供試プログラムを追跡（trace）することができる．いずれの場合にも供試プログラムが他のプログラムに被害を与えないように，使用アドレスの範囲を限定する金物あるいはプログラムが必要である．

（2） 予め入れてあるプログラムに，手動でデータを入れて優先的に問合せをすること．これは IBM-7070 などでもすでに試みられている．

（3） 機械の監視．監視用ルーチンを定期的に割込ませれば，レジスタだけでなく，任意の記憶場所を見ることができる．

（4） 従来オフ・ラインの機械で取扱われていた仕事を僅かの計算機時間を割くことによって，計算機に兼業させ，金物を節約し，かつ操作の融通性を増すことができる．

（5） 計算機制御などの実時間動作に優先的に計算機を使用する傍ら，科学計算，事務データ処理などの全然別の仕事にも使える．

6. 計算機の多重化

時分割とは逆に，一つのプログラムを複数個の計算機で処理して，その速度を上げようとする行き方がある．NBS の Pilot[3,4] はその代表的なもので，主計算機，“雑用”計算機および入出力計算機の三つが有機

的に結合されている．有機的というのは，単にこれらの三つが仕事を分担するだけではなく，

（1） それぞれの記憶装置の間でデータをやりとりする．

（2） 一方の計算機から他方の計算機に命令を植付ける．

（3） 相手の計算機の状態を調べ，その結果によって自分が飛越しをする．

（4） プログラムの特定の点で相手の計算機に準備完了の信号を送る．

などということをやりながら，仕事を進めて行くわけである．

計算機の基本回路や記憶素子の速度が限界に来ると，それ以上計算速度を上げるには，装置を複数個並列にするより方法がないわけであるが，単に装置を増すだけではなく，このように演算用，“雑用”，入出力用と機能別に専門化して効率を高め，さらに各装置を一応独立させて融通性をもたせようというのは，方式設計上極めて興味の深い行き方であろう．

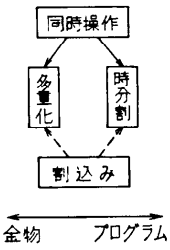
一方，現在の大形計算機が，大抵の問題ではその一部だけしか使われていないという点に注目し，これを小さな同形の計算機に細分し，必要に応じてこれらを有機的に結び合わせるという方向に進んだのが，Ramo-Wolldridge の RW-400[15]である．これは多数の計算機（記憶容量 1,024 語），バッファ記憶装置（容量 1,024 語），外部記憶装置および各種入出力装置を，マトリックス状の交換回路網によって適当に交換接続するもので，あるときには互に独立した多数の小形計算機となり，あるときには有機的に結合した一大計算組織になるように計画されている．各計算機には Stretch の方式によく似た割込み機能が設けられており，計算機同志が互に割込み合うことによって，また交換接続をプログラムで制御することによって，全体の有機的な関係が保たれる．

前述の ETL Mk-4 B は，既設の Mk-4 A と結合される予定であるが，その結合は Mk-4 B の記憶装置を Mk-4 A からも使えるようにすることと，互いに割込み合うことによって行う．

この種の計算機の多重化については，まだどこでも充分な経験がなく，今後に残された面白い問題の一つである．

7. 計算機の高速化

プログラムを並列に実行するには，前述のように時分割と計算機の多重化の二つの方法があり，割込み技術はこれらの効果を高めるために，そのいずれにも利用される．これらの関係を示すと**第7図**のようになる．時分割に種々のレベルがあることを述べたが，このレベルが下がるほど多重化に近くなってくる．



第7図 諸方式間の関係

これは見方を変えれば，できるだけプログラムに頼るか，金物に頼るかということになる．高いレベルの時分割を使用することにより，金物は節約でき融通性は増大するが，それには金物の速度が充分大きくなければならない．磁気テープや外部記憶装置としての磁気ドラムは，遅い計算機で時分割制御できるほど充分低速ではない．かりに磁気テープの速度を 50 kc とし，10字よりなる1語ごとに割込み，割込むごとに4ステップの命令を実行し，各命令の実行に 20 μs を要するとすれば，計算機時間の 25% がただ1台の磁気テープに費されることになる．最上位の時分割を行うには各命令の実行に要する時間を μs あるいはそれ以下にすることが必要であろう．

電気試験所で現在設計中の ETL Mk-6 には時分割方式を徹底的に採用する方針であるが，これには 5 Mc 2相のクロック・パルスに同期する基本回路を用い，加算器にはマンチェスタ大学で開発された高速桁上げ回路[16,17]を利用して，200 mμs で加算を完了する．また主記憶装置はサイクル時間 2 μs の磁心マトリクスであるが，他に呼出時間 100 mμs のトンネル・ダイオードによる記憶装置および同じ呼出時間の固定記憶装置をもつ．

計算機の価格は速度に比例して高くなるものではない．ETL Mk-4 A と Mk-6 の速度は 1：500 であるが，製作費は 1：10 に過ぎない．一方やれる仕事は時分割の利用により，ほぼ速度に比例するから，ある限度までは計算機は速ければ速いほど得ということになる．

なお，全体としての計算能力を高めるために計算機を多重化するという行き方には大して魅力はないが，複雑で大きな金物を複数個の簡単な計算機に分けて，金物としての構成を簡単化するとともに融通性をもたせるという考え方に立てば，計算機の多重化も捨てたものではない．バッファ記憶装置のような能力のない金物をやたらに増やすことは無駄であるが，計算機が

複数個あるということになれば，それだけの利点は得られよう．Mk-6 では浮動小数点表示の仮数部と指数部を別の計算機によって計算し，指数部計算機には“雑用”をも行わせることを検討中である．

参考文献

1) J.P. Eckert: “Univac Larc, the Next Step in Computer Design”, Proc. EJCC, p. 16 (1956).
2) W.F. Schmitt and A.B. Tonik, “Sympathetically Programmed Computers”, Unesco/ICIP/B. 2.18 (1959).
3) A.L. Leiner et al.: “PILOT-A New Multiple Computer System”, J.A.C.M., **6**, 313 (1959).
4) A.L. Leiner et al.: “Concurrently Operating Computing System”, Unesco/ICIP/B.2.17 (1959).
5) 高橋秀俊，後藤英一，他：“パラメトロン電子計算機 PC-1 について”，通信学会電子計算機研究専門委資料 (1958).
6) M.V. Wilkes and D.W. Willis: “A Magnetic-Tape Auxiliary Storage System for the EDSAC”, Proc. I.E.E. **103** pt B, 337 (1956).
7) M.V. Wilkes and J.B. Stringer: “Microprogramming and the Design of the Control Circuits in an Electronic Digital Computer”, Cambridge Phil. Soc., **49**, 230 (1953).
8) C. Strachey: “Time Sharing in Large Fast Computers”, Unesco/ICIP/B. 2,19 (1959).
9) 淵一博，西野博二：“入出力用計算機 ETL Mk-4B の方式”情報処理，**1**, 16 (1960).
10) J.W. Forgie: “The Lincoln TX-2 Input-Output System”, Proc. WJCC p. 156 (1957).
11) F. P. Brooks, Jr.: “A Program-Controlled Program Interruption System”, Proc. EJCC p. 128 (1958).
12) 和田英一：“Overlapping Computation— PC-1 の割込み機能”，通信学会オートマトン研究専門委資料 (1959).
13) 金田弘，他：“NEAC-2203 電子計算機の割込み”電気3学会連合大会予稿 375 (1960).
14) S. Gill: “Parallel Programming”, Computer Journal **1**, 2 (1958).
15) R.E. Porter: “The RW-400, A New Polymorphic Data System” Datamation **6**, No. 1, 8 (1960).
16) T. Kilburn et al.: “Parallel Addition in Digital Computers; A New Fast ‘Carry’ Circuit” Proc. I.E.E., **106** pt B, 464 (1959).
17) 高橋茂，西野博二：“マンチェスタ大学の高速桁上げ回路の追試と二，三の考察”，情報処理，**1**, 23 (1960).