時分割方式における記憶装置*

石　井　善　昭**

1. 緒　　言

時分割による演算方式は元来入出力装置の速度が内部演算速度に比べて非常に遅いのを補うために計算機構を時分割で使用し，計算システムとしての処理能力を高めるために開発された技術である．

このような時分割の技術は複数個のプログラムを一つの計算機にかけて同時に処理させる多重プログラムを可能にするもので，これは時分割処理を金物に多く依存するものと，プログラムに多く依存するものとがある．前者としては複数個のプログラムに割り当てられたシーケンスレジスタを切り換える回路をもっていて，内部演算に関してはプログラムの1ステップ実行ごとに切り換えが生じるもの，あるいは内部演算に関しては，一定時間ごとに他のプログラムに切り換わる方式のものなど，種々の方式がある．

また，入出力終了，入出力の誤り，外部からの割込信号，内部演算中の誤りなどが検出されると，自動的に実行中のプログラムが中断され所要のルーチンへ切り換わる謂ゆる automatic interruption の機能をもつ計算機も金物の要素が強いプログラミング方式といえよう．

計算機によっては busy test の命令をプログラムの随所にばらまいておいて，入出力の多いプログラムの入出力時間を所要のルーチンに切り換えて併行処理を行なわせるかなりプログラムの要素の強い多重プログラミング方式もよくとられている．

金物で各プログラムチャンネルを一定条件のもとで次々に切り換えていくような多重プログラミング方式がとられた場合，中断が生じたとき必要な情報が記憶装置に格納されず演算ユニットとかレジスタに残っていることがある．この場合にはレジスタの退避の問題が生じ，特に一番地方式の計算機ではやっ介な問題になる．

これら併行処理とは逆に一つのプログラムを複数個の計算ユニットで処理し，計算システムとしての処理能力を高めようとするものに multicomputer[1,2]がある．NBS の PILOT[3], Ramo-Walldridge の RW-400[4,5] などがそれで，これらは一つの計算機の中に独立に働き互に融通性をもつ複数個の計算ユニットをもっている．

これら多重プログラミング，多重処理の技術の採用によって互に独立なプログラムを併行処理しようとする場合，全体のプログラムを監視し，調整する supervisory program が必要になり，これに関連して，memory protection, relocation などの問題が生じてくる．

以下これら時分割方式における諸問題のうち，記憶に関したものについて具体的に述べる．

2. 時分割におけるコントロール記憶

時分割による多重プロダラミング方式の計算機には種々の形式があるが，複数個のプログラムを計算機の金物が時分割で切り換えていく方式がとられる場合，時分割制御に特有な記憶装置として，各プログラムチャンネルに対応したスペシャルレジスタ記憶を必要とする．

このスペシャルレジスタ記憶は各々そのチャンネルに入力されたプログラムを独立に実行させるために必要な演算制御レジスタ，シーケンスレジスタ，インデクスレジスタなどをすべて含んだコントロール記憶であって，一般の working memory と異なり多重プログラム制御のもとで働く．

このコントロール記憶はプログラムチャンネルの数が大きくなればなる程，容量が大きくなるから，フリップフロップなどでなく通常内部記憶の一部分を割り当て，実行中のプログラムが中断され，再びそのチャンネルに戻ってくるまで，中断時のプログラムの進行状態を保存するための制御用レジスタをすべて含んでいる．

UNIVAC-1107 ではこれらの制御を高速に行なうためにコントロール記憶に磁性薄膜を用いている．

コントロール記憶の動作を H-800[6]を例として説明すると，コントロール記憶は磁心記憶で，プログラム

---

* Storage in Time-Sharing System, by Yoshiteru Ishii (Nippon Electric Co., Ltd. Tokyo Japan)

** 日本電気株式会社

制御を行なう32個のレジスタグループ8組で構成されている．

いま一つだけのプログラムがどれかのプログラムチャンネルに load されている場合は，単にそのスペシャルレジスタグループのみが ON になっていて，そのプログラムだけが実行されるが，二つ以上のプログラムを load した場合，たとえば #1，#2，#8 のスペシャルレジスタグループが ON の状態にあると，#1 のプログラムが1ステップ命令を実行したら，#1 のスペシャルレジスタの内容はそのままの状態にしておいて，マルチプログラムコントロールは #2 に制御を切り換える．

この場合 ON の状態にあるスペシャルレジスタグループのみが走査され，いまあるプログラムが busy の状態にある入出力チャンネルを指定した場合は，そのスペシャルレジスタグループを一時“ストール”状態にし，次の ON になっているスペシャルレジスタに切り換えられる．

その入出力チャンネルの動作が完了すると，“ストール”状態にあるプログラムは動作状態に戻り実行されることになる．

3. 入出力バッファ

入出力装置の動作速度は計算機の内部演算速度に比べると非常に遅い．

したがって入出力装置を直接計算機の内部記憶に結合させると，入出力装置が動作している大部分の時間計算機はインターロックされていて，計算機の処理能力は入出力速度に押えられてしまう．

そこでこれら入出力装置に一時記憶を付加して入出力装置と一時記憶との間の情報の授受をオンラインで行なわせれば，計算機本体は一時記憶と主記憶との間の情報の移送のときだけ占有されるのみであり，他の時間は入出力動作と併行して他の計算に利用することができる．このための一時記憶が入出力バッファ記憶であり，入出力装置と内部記憶装置との間におかれ，入出力装置と計算機内部記憶との間の情報の移送が行なわれるときは，必ず一たんこの入出力バッファに入る．バッファ記憶の容量[7]は通常入出力装置の機械的動作が始動停止できる1くぎりの桁数をもつことが望ましい．このときは，仮りに入出力装置の動作に併行して実行される他のプログラムの実行時間が予想以上に長くなることがあっても，入出力装置の速度低下を招くだけで，致命的な情報のミスを招かないからである．

すなわち紙テープ入出力では1文字，カード入出力ではカード1枚分，高速度印字では1行分のバッファ記憶がこれにあたる．

カード入出力，高速印字のようにある程度まとまった容量の入出力バッファ記憶には磁気コアが多く用いられる．磁気テープはこの論旨でいくと，1ブロック分の入出力バッファが望ましいのであるが，この場合に1ブロックの桁数が大きくなること，磁気テープの高速性等により事情が異ってくる．

NEAC-2204[13] のごとくバッファ記憶に低廉な磁気ドラムの高速アクセストラックを流用している場合は1ブロック分のバッファをおいてもひきあうが，そうでない場合は1ブロック分のバッファをもたせることは不利になる．このような場合普通入出力チャンネルに2語のバッファ記憶をおいて，片方の1語が，磁気テープと情報の授受をしている間に，他の1語のレジスタと内部記憶の情報授受および他のプログラム実行を行なうようにしている．

1ブロック分のバッファ記憶をもつときは，他のプログラム実行時間はかなり長くとれ，またその時間が長すぎてもテープ情報を失うことがないが，2語のバッファ記憶のみをおくときは，その時間に極めて厳格な制限があり，この時間を他の命令に割り込ませることはあまり意味がなく，かえってプログラマにかかる負担が大きくなる．したがってこの時間は他の磁気テープの情報移送のみに使用されるようになっていることが多い．

4. レジスタの退避[8]

時分割による演算でプログラムで中断が生じたときに必要な情報が内部記憶に格納されずに残る場合がある．これは2番地方式，3番地方式が採用されている場合は比較的少く，NEAC-2204[8,13] では全く考慮する必要がない．その他の計算機でも乗除算命令，ジャンプ命令，インデクス関係命令などのみが問題となることがある程度で，この場合は他のプログラムの走査を禁止して同じプログラムの次の命令を実行するようにしている．

しかし一番地方式の計算機では演算結果は通常演算レジスタに残るため，中断が生じたときに演算レジスタに残されている情報は，一たん定った記憶場所に移し，再びそのプログラムに切り換わったとき，その情報を元のレジスタへ戻し，処理させなければならない．

したがってN個の独立なプログラムを実行させる場合には，N組の退避用記憶場所を必要とし，かつ退避のためのプログラムを，収容する記憶場所を必要とする．

これらの記憶容量及び退避用のプログラム実行時間を要することを考えると，時分割方式では，2番地方式，3番地方式のような多番地方式が都合のよいものであるといえる．

LARC[9]はこの損失時間をできるだけ小さくするため，自動的に中断せず，プログラム的に都合のよいプログラムステップで，切り換えて行なうようにしている．

5. Supervisor

計算機が大形化し，より高速化し複雑化してくると，人間の介入をできるだけ避けるため，自動的にプログラムを実行させ，その進行状態を監視するプログラムシステムが要求されてくる．

このプログラムがsupervisorである．supervisory programの機能としては

1) Scheduling
2) 並列処理における優先権の調整
3) プログラムの relocation
4) memory protection
5) 誤りの処理

などがある．

これらsupervisory programはかなり大容量の記憶を必要とするので，大形，かつ高速の計算機に通常適用される．

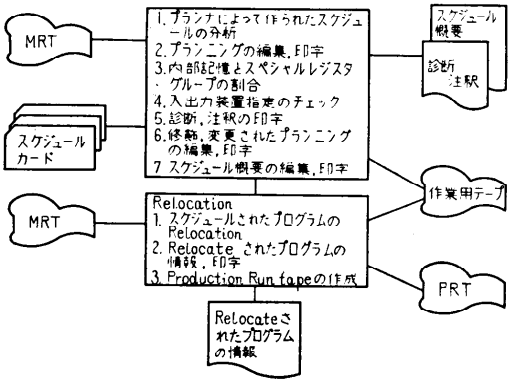
以下，これらにつき説明する．

5.1 Scheduling

多重プログラミングにより複数個のプログラムを並列処理する場合に，これらを自動的に行なわせるにはプランナによってスケジュールされたとおりに処理されるよう各プログラム間の整理が必要である．

この段階が Scheduling であり，これを H-800 について説明すると Supervisory Program の Scheduling Run は，プランナによって作られた各プログラムについてのスケジュールを分析し，計算機の利用できる記憶容量，入出力装置などが両立するかどうかをチェックし，もし，両立しなければ誤りを指摘し，Production Run Tape を作る．この Production Run Tape からプログラムを入力し，スケジュールに従ってそれらのプログラムを ON にしたり OFF にしたりする．

この Scheduling の機能をさらに具体的に示したのが**第1図**である[10]．



第1図　H-800 における Scheduling の機能

Scheduling Run の入力は，ユーザプログラム並びに Supervisory Program が入っている磁気テープ MRT と，プランナがプログラムの並列処理に関するスケジュールを行なったカードからなっている．

この MRT のユーザプログラムには relocatable information がついており scheduling は assignment と relocation の動作を行なう．

それらの動作は第1図に示すとおりである．

この scheduling は3本の磁気テープを用いて行なわれ，最終的には Production Run Tape が作られる．

5.2 優先権の調整

MIT が IBM の 7090/94 による CTSS[11] のために用意した Supervisory Program について優先権の調整を説明する．内部記憶はA記憶とB記憶に大別される．

Supervisor Program はA記憶に格納されていて，

1) コンソールのためのすべての入出力を取り扱うこと
2) foreground job と background job のスケジュールをたてること．
3) 一定時間ごとに切り換えられるようにプログラムを一時的に格納したり戻したりすること．
4) background job についてすべての入出力をモニタすること．
5) すべての foreground job についてモニタすること．

などを含む膨大な supervisory program である．

supervisor は q 秒ごとにコンソールの入出力を行なうために B-記憶で進行しているプログラムを中断する．

コンソールからの入力が制御のものでなければ，ユーザのプログラムで読取られるまで supervisor 用のコアバッファにおいておく．

入力が制御のものであれば，優先して supervisor は磁気ディスクから必要な情報をもってくる．

この最も優先順位の高いものを除くと，時分割プログラムはあらかじめ決められた q の倍数の時間ずつ実行する．各プログラムの適当な時間の終りに supervisor が次のどのユーザのプログラムを進めるべきかを決める．それから磁気コアに現在あるプログラムを次のユーザプログラムが磁気コアを使用できるように，磁気ディスクまたは，磁気ドラムに移すべきかどうかを決める．supervisory program は各ユーザプログラムのコンソールのための入出力バッファをもっているほかに，各ユーザプログラムの状態の記録をもっていなければならない．ユーザプログラムの状態には次のものがある．

1) working
2) waiting command
3) 入出力待ち
4) dormant
5) dead

1) の working はプログラムが次に実行されるときは，常に続行してよい状態．2) の waiting command はユーザプログラムがコンソールからの command を完了したとき，3) の入出力待ちはプログラムがコンソールの入出力のために一時待合わせている場合，4) の dormant はプログラムを中断し supervisor にプログラムを戻したが，計算機の内部状態はテストや修飾のためにとっておく場合，そして 5) の dead はプログラムが終了した場合である．

supervisory program は，これらの状態を監視し制御しているが，このような大規模な supervisor には，膨大な記憶容量を必要とする．

5.3. Relocation

時分割方式では一つの内部記憶の中に複数個のプログラムが収容されるのが一般であるが，これらのプログラムは独立にプログラムされており，それらをそのまま load した場合，プログラムの格納番地，番地指定などが重複したり，入出力装置の指定が重複したりして混乱を起す場合がある．

また，これらのプログラムは許容される記憶場所にできるだけコンパクトにつめることが望ましい．

これを処理する手法が Relocation であって，始めから relocate する場所を決めて行なう static relocation と，計算段階中にその結果により適宜 relocation する dynamic relocation が考えられる．

現在実用されているのは static relocation である．さらに多重プログラミングにおける relocation には，記憶の relocation と入出力の relocation がある．

記憶の relocation はまず命令の格納番地に対してはプログラム上で一定の数を命令の格納番地に加算することにより行なわれる．

命令における指定番地の変更に対しては machine coding のとき，あるいは assembler, compiler でプログラムが機械語の命令に変換される前の assembling, campiling の段階で，relocable information をその命令につけて，それにより relocation を行なう場合が多い．

relocation の機能は，普通 supervisory program がもっており，プログラムの loading の段階において relocation を完了させる．

CTSS では 256 語を一つの記憶単位として，すべての番地を修飾する 7 ビットの relocation register をもっていて，プログラムを実行するときに relocation register のブロック数を各番地に加える方式をとっているが，これもやはり supervisory program のもとで行なわれる．

このように supervisory program で中断されたプログラムは，必要があれば relocation register を適当に再調整するだけで記憶内容を移動できる．

5.4. Memory Protection

完全に debug の終っていないプログラムでは，実際に働かせると予定外のところに制御が移り暴走することがある．

これらの予定外の領域にプログラムとして働くものがあり，そのプログラムが実行されると本来の計算が行なわれず，時にはそのプログラムをこわしてしまう恐れがある．

もし，この場合予定外の領域に存在するプログラムが supervisor などの場合には，特にその影響は重大であり，プログラムが暴走しても，そのプログラムが乱されないような protection がなされることが望ましくなる．memory protection を金物で行なわせる方法と，プログラムで行なわせる方法とが考えられる．

CTSS では supervisory program とユーザプログラムとの間の記憶に関するお互の干渉を避けるために次のような方法をとっている.

すなわち前述したとおり記憶装置に256語を一つの記憶単位として取り扱い，その記憶単位について7ビットの protection register を2組もっており，これがそのユーザプログラムに許容されるブロックナンバーの範囲を示す.

計算機が normal mode にあるときは，常に supervisor はユーザプログラムに移る直前に計算機を special mode に切り換える.

special mode では protection register の範囲外の番地が指定されると，supervisory program へ移り処理される.

このように protection のきっかけを金物からもらい，supervisor にその機能をもたせているものに対して，supervisory program とか種々のルーチンを消されることのない固定記憶にあらかじめ格納しておいて保護しようとしたものに Ferranti の ATLAS[11] がある.

ATLAS の固定記憶は最大容量 260 k 語，アクセスタイム 0.2 μs のフェライトプラグの wire mesh 記憶で，記憶の内容はプラグがあるか，ないかによりかえることができるが，これには時分割に基づいて周辺の動作を扱うルーチンと supervisory porgram が格納されている.

このように固定記憶にあまり変更の必要のないコントロール・プログラムルーチンを入れて，protection を行なうのは最も確実な方法であり，望ましいものであると考えられる.

6. 結 言

以上，時分割方式における諸問題のうち，記憶装置に関係すると思われるものについて説明したが，計算機によっては始めから必ずしも時分割を考慮してつくられたものでないため，ほとんど全面的にプログラムに依存するものがあるが，やはり memory の relocation などはともかく，memory protection，アドレス方式，レジスタの退避，プログラムの切り換えなどは，金物的にできるだけ処理されるのが望ましいと思われる．このためには固定記憶，コントロール記憶に適用される高速記憶など，今後の記憶装置の発展に期待するところが大きい.

参考文献

1) Walter F. Bauer: Why Multi computers?, Datamation. September. 1962, pp. 51～55
2) G.M. Amdahl New Concept in Computing System Design, Proc. IRE, Vol. 50, No. 5, 1962, pp. 1073～1077
3) A.L. Leiner, et al: Pilot-A New Multiple Computer System, J. ACM 6, 313, 1959 pp. 313～335
4) R.E. Porter: The RW-400 A New Polymorphic Data system, Datamation 6, No. 1, 8, 1960
5) R. Perkins, et al: Programmed Control of Multi-Computer Systems, Unesco/ICIP 1959 pp. 545～549
6) N. Lourie, et al: Arithmetic and Control Techniques in a Multiprogram Computer Proc. EJCC, 1959, pp. 75～81
7) 石井: 時分割演算方式における待合せ及び改善率について，情報処理，Vol. 3, No. 1, Jan. 1962
8) 石井: 時分割による計算機の研究，日電電機資料 114, Feb. 1961
9) W.F. Schmitt, et al: Sympathetically Programmed Computers Unesco/ICIP 1959, pp. 344～348
10) Scheduler and Run Procedure, Honeywe 800 and 1800 Executive Manual.
11) F.J. Corbató, et al: Compatible Time-Sharing System-A Programmers Guide, The M.I.T. Computing Center, 1963
12) T. Kilburn, et al: The ATLAS Supervisor Proc. E.J.C.C. Vol. 20, Dec. 1961, pp. 279～294
13) 金田，石井，斎藤，山田: NEAC-2204 電子計算機システムについて——電子計算機専門委員会資料 (1961-3)

(昭和38年11月22日受付)