計算機の時分割使用について*

穂　坂　　衛**

1. まえがき

最近，計算機の time sharing の用い方や，computer utility や，あるいは，MIT の MAC 計画がしばしば計算機関係者の間で話題になるようになってきた．昨年9月の当学会の月例会で筆者のこれらについての簡単な報告にも割合関心が集ったとのことであるので，それを補足して会誌における解説としたい．始めに簡単な実例を示し，ついで，従来の研究や現在の各方面の計画などに言及し，最後に MIT の MAC 計画のいままでの方式や結果，さらに今後の動きについて触れる．

従来の計算機の主要な使い方は，いわゆる一括処理方式であって，高価な計算機を遊ばせないために，用意された使用者のプログラムを連続的にかけ，使用者の介入は許されなかった．したがって，計算機にとっては誠に効率がよいが，使用者には必ずしも幸福を持たらさず，問題を解く意志をもったときから，満足のいく結果を得るまでの時間が長く，手間も多かった．また，人の考えに合わせながら開発を進めていく種類の仕事は非常にやりにくかった．

計算機の時分割使用方式は，人と計算機とが直接会話のできることが，始めの目的であり，つづいて，計算機のもつ高度の能力を大容量の記憶を共同に利用できる多数のプログラムを，個人の仕事のために，各人が直接に利用できることに重要な点があることがわかってきた．端末のコンソールより，計算機に指令を出し，各種のソフトウェアで問題を解くプログラムを作ることを援助してもらったり，でき上ったプログラムの実行を要求したり，途中でそれらを変更することも，前もって定めておく必要はない．しかも多数の使用者が関係なく，同時にこのような使用法ができるのが特徴である．普通に用いられる time sharing という言葉だけでは，この一部の意味しか表わしていない．MAC と MIT で称するのは，Multi-Access Computer あるいは Machine Aided Cognition の略であるが，これは hard ware の形式と使用者の希望を表わしたものであろう，最近は，Multics (Multiplexed Information and Computing Service) などともいっているようである．

非常にルーチンな仕事を計算機に与えて，とにかく高度に多量の出力データを得るのなら話は別であるが，人が考えて問題を解くのに，高性能なハードウェアとソフトウェアをもつ計算機を利用しようというのであれば，人の時間の大部分は，考えることと人の機械的な操作との混合であるから，計算機の介入する時間は少なくてすむ．そのため多くの使用者が中央の高性能な計算機を共有して用いることができ，それによって，計算機も使用効率の低下を防ぐことができる．

現代の計算機の大部分は，そのハードウェアとソフトウェアにおいて，このような用い方を意識して開発されてきたのではないため，time sharing に対しては不完全でまた能率も十分よくはない．また，time sharing の方式自身も確立されてはいず，今後の研究開発に待つところが多いし，このような用い方に批判がないわけではないが，これは，計算機の実時間使用における大きな流れの一つであることには間違いのないところであろう．

2. 実　　例

現代の time sharing system を詳しく論じる前に比較的単純な GE/Dartmouth システムの使用例をあげ，端末のタイプライタ入出力の感じを示し，そのあと各方面における time sharing computer への努力を概観する．

第1図は，端末のテレタイプの入出力記録の一部である．使用者はキイを押し，ダイヤルして，どの端末からかを中央の計算機に通知し，HELLO とタイプすると，中央から USER NUMBER はときいてくる．S 1000 と使用者が自分のコードをタイプすると，そのあとどのプログラムシステムを使うのか，この問題は新か旧か，新ならばその名前は，を次々に聞いてくる．それらに答え終ると，READY といってくる．使

* On Time-Sharing System, by Mamoru Hosaka (Inst. of Space and Aeronautical Science, Univ. of Tokyo)

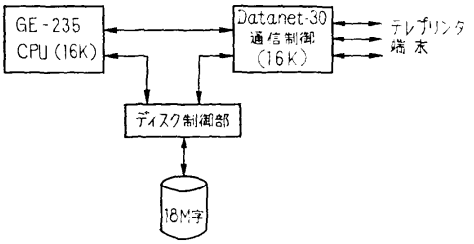
** 東京大学宇宙航空研究所

```
HELLO
USER NUMBER－－S 10000
SYSTEM－－BASIC
NEW OR OLD－－NEW
NEW PROBLEM NAME－－DEMO
READY.
10 FOR I=1 TO 10 STEP 2
20 LET Y=SIN (COS(TAN(ATN(LOG(EXP(I)
                                    )))))
30 PRINT I, Y
40 EN ←← NEXT I
50 END
RUN
DEMO  8:25  THURS.  4-29-65
1              ·514395
3             -·836022
5              ·279873
7              ·68449
9             -·790197
TIME:  0 SECS.
LIST
DEMO  8:25  THURS.  4-29-65
10 FOR I=1 TO 10 STEP 2
20 LET Y=SIN (COS(TAN(ATN(LOG(EXP(I)
                                    )))))
30 PRINT I, Y
40 NEXT I
50 END
CATALOG
SAVED PROGRAMS, USER NUMBER S 10000
THURS. 4-29-65   TIME: 8:26
NAME     GEN      PLOT     PI
SINE     PRIME    SINEA    MORT
MORTI    GRAPH    BORST    LSAJOS
MATINV   SINE 40  HIST 40  FLIGHT
ALPRIM   ELLIPS   HISTO    INFORM
SPIRAL   SPIRL 2  LES 34   HYPERB
LES 15   NORMAL   CIRCLE   LES 23
TAX      CAR      PHILOS   HELP
BEST     AD
BYE
*** OFF AT  8:27
```

第1図　実　　例

用者は，ステートメントを書き込み，書き誤まればその場で取り消し（40 の EN ←← は，EN を取り消している），訂正できる．RUN というコマンドでいまのプログラムが実行され，結果が印刷される．8時 25 分から実行され，0.5 秒以下で終了したことも示される．いまのプログラムを LIST することや，既に過去に SAVE したプログラムを CATALOG していることも示され，BYE といって，8時 27 分に仕事を終了している．この他のコマンドに，RENAME, SCRATCH, STOP などもあり，名前をつけかえたり，くずかごに投げ込んだり，仕事を中止させたりすることができる．

この機械の構成は，**第2図**に示すように GE-325 と Datanet-30 の通信制御装置およびディスクから成り，GE-325 は主として使用者のプログラム処理を行なって従計算機となり，Datanet-30 は，30 個の端末との入出力および従計算機の使用スケジューリングを行なう主計算機となっている．



第 2 図

これに似た，かなり限定された種類の言語を用いる time sharing の計算機システムには，この他に，PDP-1[1]，IBM-7040[2] および RAND の JOHNIAC[3] を基本に用いたものが開発された．PDP-1 は，その後 PDP-6 に改良されて売り出され，IBM-7040 の場合も Quicktran の名称の言語で，Console を貸す商売を始めているようである．

一方，JOHNIAC は，1950 年代の初期に作られた真空管式で，4K 語のコア・メモリをもつが，非常に原始的な構成の機械である．それに 12K のドラムと Communication Control をつけ，JOSS と称する time sharing を作りさげた．RAND Corp のように大型の最新の計算機をもち，しかも専門家の多い所でも，100 人を越す使用者は JOSS の速度に不満をもたず，それらの使用者の要求は，もっと多い JOSS の使用時間と，その Storage の増加であると報告されている．さらに，General Purpose の time Sharing は，System Dev. Corp. における Q-32 を中心としたもの，および IBM Advanced System Development Div. における 7090 を中心としたシステム[5] が報告されている．

この他に，最近各メーカーは新しく time sharing 用の計算機を続々と発表している．たとえば GE-645 は，MIT の MAC 計画の次の候補となり，これは，Bell Telephone や Martin Co., Hughes Aircraft Co. などでも用いられることになり，IBM 360 は，Model 67 を time sharing 用に発表し，Bell Tele-

phone, MIT Lincoln Lab., MIT Computation Center, Carnegie Inst. of Tech., その他に設置するとのこと，また，前述の RAND の JOHNIAC は，PDP-6 におきかわり，Scientific Data Sys. 940 は，California 大学に，またその Lawrence Radiation Lab. は，CDC 6600, PDP-6, IBM 7094, Stretch, CDC 3600 を将来まとめた system を考えるとのこと，この他 CDC 3300, 3500, PDP-6, Burroughes B-8500, Honneywell 8200, その他続々名のりがあがっている．わが国でも，幾分その動きが伝わったか，二，三の所で検討し始めたようである．

いままで述べたのは，主として，科学技術の面における time-sharing であったが，それ以外にも大きな応用の場がある．日本国有鉄道の座席予約システムは，約500個所の端末からの各種の要求を通信線で中央処理装置に直結し，そこでは，複雑な Multi-programming と Multi-processing の技術が，実時間での大容量ファイル処理のために使われ，個々の要求を高速に処理して結果を端末に送り返えす．多い日には，25 万件に達する要求を，同時に処理するだけではなく，同時にマネイジメントや統計上の各種の報告もつくり出している．いくつもの企業間で time sharing を行なっている例では，アメリカの Key data という会社は，time sharing で酒類卸商達のために，発送および在庫管理を 24 個所の端末をもって行なっている．この会社は，今後端末を数百にふやし，中小企業のために各種のサービスを行なうことを考えている．この他病院における time sharing も開始されたようである．

これらよりさらに大きな目標のもとに，time sharing に非常な情熱と努力を重ねてきている所は，MIT であり，最近は Project MAC という名で呼ばれ，人々の関心を引いている．これについては，次節以下で，Control の方法，機器，Software，各種の問題点に触れてみよう．

3. MAC 計画

MAC というのは，Multiple Access Computer あるいは，Machine Aidei Cognition の略語である．後者の意味は，人と計算機が実時間で協力し，両者の能力を最大に用いて問題の解決をはかろうという意図を示しているのだそうである．Cognition そのものの意味は，human way of thinking とのことである．この協力を行なうためには，計算機システムは，多数の使用者が，同時に物理的にも，知識の上においても，容易に Access できるものであることが必要であり，現在の MAC システムはその第一歩である．これは，現在，次々に拡大が行なわれている．拡大は，Hard, Soft 両者にわたっており，中央の計算機自身も 7094 より GE 646 に変換されていく最中である．したがって全体の正確な形は捕えようがないが，最近の資料や訪問の際に得られた情報をもとにして解説を行なうことにする．

MIT において time sharing の考えと，その実現のための計画が発生したのは 1960 年前後であり，MIT の計算センターの 7090 にいくつかの改造を行なって，3 台ほどの入出力タイプライタを取り付け，最初の time sharing のデモンストレーションを行なったのは，61 年の 11 月である．これについては，62 年の SJCC[6] に報告されている．さらに引き続き，このシステムを従来の計算センターの仕事と両立させること，console の数の増加，通信方法の整備，コアメモリの増加，disc, drum の付加，7750 channel や各種の機能の追加が行なわれ，Compatible Time-Sharing System (CTSS) と呼ばれるものになった[7]．始め MIT の計画であったこの仕事に，1962 年より国防省の Advanced Research Project Agency が興味を示し，資金を出すようになった．

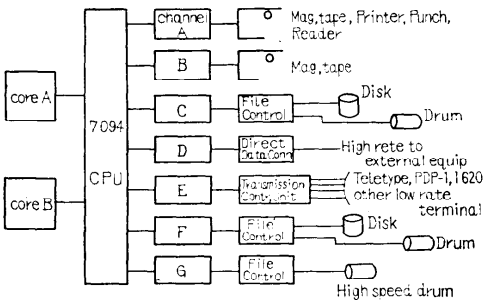
MAC Computer System は，この CTSS の延長にあるわけで，計算センターにあるシステムとほとんど同じものを，もう一組 MIS の構内につくり，仕事は両者共同して進めている (1963 年 11 月)．MAC の思想は，1961 年 4 月の Prof. John McCarthy の講演に述べられている．これは，MIT における計算機の将来の必要性調査委員会の結論を代表するもので，Computer の power を，あたかも電力の power を分配するように，場所，時間，あるいは量を問わずに分配できるような utility の性格を考えている．これは電力が各種の physical work に役立つと同様 Computer power を，広範な intellectual work に役立たせることであり，いままでの計算機は，初期の蒸気エンジンのようなもので，この power は，各個人単位まで分配できなかった．

このように個人の必要に応じた power を与えるということに止まらず，各種の public procedure, data, program などを容易に利用でき，個人のファイルの貯蔵，再生，また大勢の人の協同の仕事をこれを通して実行できる手段を提供できることも意図して

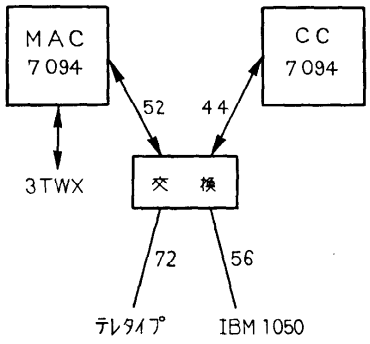
いる．このように大変な目標をもっている MAC も現実には，まだ，Multiple Access Computer が主で，いままでの大形計算機で用いられた batch monitor の欠点を補い，man-machine の communication をよくして，能率のよい debug の技術を用い，プログラム製作に役立せているが，さらに decision や real time management, gaming などの問題，sociological experiment, teaching machine, language learning, library retrieval, text editing, algebra manipulation, graph manipulation 等々のもう一つの MAC の意味への努力もかなり行なわれつつあるようである．

4. 装　　置

MAC そのものは，特定な機械とは限らないわけであるが，現在用いている計算機は 7094 が中心で**第3図**のようになっている．MIT Computation Center および Project MAC の計算機は，**第4図**で示されるように外部と通信線で結合している．7094 には，幾つかの special feature が付加している．おもなものは



第 3 図



第 4 図

（1） Core storage interval time clock

（2） Memory protection and relocation registers

（3） Channels

A. Printer, punch, reader, chronolog clock

B. Tapes

C. Disc and Drum Storage

D. Direct Data input and output

E. 7750 Communication channel

F. High Speed Drum

（4） 2×32 K Core Storage modules (Core A, Core B)

Core B には，user's program, core A には，supervisor, 9 M words/disc, 185 K words/drum. core の 32 K を全部移動するのに，disc では 2 秒，drum では 1 秒，高速ドラムでは 0.25 秒，disc は，private files, public commands, compilers を貯え，drum は，core の退避場所，7750 には，remote typewriter その他の低速通信線や他の計算機などへの高速通信線が結合，7750 は一種の stored program の computer である．

（5） 全システム設計の方針は，(1) 7094 に全体の control の集中化．(2) 最大の interraction rate の確保．(3) 使用者は，Supervisor に仕事の要求を出すだけで，詳細は，Supervisor subrontine が行なう．(4) Supervisor は context-free であること．すなわち，command をさがし，load し，start させるだけ．

5. User, User's Program と Supervisor との関係

このシステムの使用者の identification と use previlege の情報が，Master File Dictionary の中にある．使用者がシステムに入ってくるとすると，user number が与えられ，scheduling algorithm に支配されるようになる．user は，つぎの各種の state の間を移り変わる．(0) Dead, (1) Dormant, (2) Working, (3) Waiting Command, (4) Input Wait, (5) Output Wait.

(0) は，prog がない状態，(1) は run できる program はあるが，run させていない状態，(2) は実行できるように schedule された状態，実行はある

時刻には，ただ一つの user だけ，(3) は (0) か (1) の状態から，user が command を type し終ると，(3) になり，scheduler が働く．そして (2) の状態にする．この command の program は，user のそれと区別はなくなる．(4) は program が user の input を要求し，user がまだ応じないとき，scheduler からはずれる．(5) は prog. が output で待たされるとき．

Supervisor の command は，public command で type から入ってくると，その名前を command directory の中からさがし，それに付属する情報より，disk から指定された名前の file をよみ出し，core の指定された場所に移し，実行を開始させる．Supervisor は，I/O Adapter, Typewriter coordinator, Disk contiol, clock trap processor, Storage allocator, Scheduling algorithm, Machine trap location などに module 化され，12 K 語の命令と各種の表から構成されて，32 K の core memory A に常駐している．また常に，これの改良や開発が行なわれており，信頼性のある働きを確保するために，新しく変更した部分に bug が発見されると，過去のものが，disk から load してこれに代わり，使用上の trouble を避けている．

User の program (Core Bにいる) が動作しているとき，どれかの user が type をたたくと，通信線，7750 を経由して trap が生じ，Supervisor に control が移り，user 番号と共に core A 内の common input pool buffer におかれ，元々実行中の user's program にもどる．0.2 秒ごとの時間がくると，control は，supervisor に移り，その間に入った各 user の input は，対応する user の input line に行く．これで user の status change がおこると，scheduling algorithm に知らされる．出力は，これとほぼ同様であるが，バッファは 7750 にある．

Scheduling algorithm が，一人の user のプログラムを止めるべきであると判断すると swap が生じる．また program の実行されている user が，途中で別の sub-routine を要求するときも故意に trap で supervisor に入る．すべての端末は，計算機と同時に結合され得るが，早い応答を得るために，ある時期に active である端末数は，現在 Supervisor によって 30 におさえられているが，これは次第に増加されていく．

6. Scheduling および Storage Algorithm Policy

(1) state (2), (3) にある user はいくつかの level の queue をつくる．

(2) この queue は，200 ms ごとにその間に発生したことで queue の順序を変えることもある．

(3) queue には，priority がある．level j のものは，level $j+1$ の queue より先に実行する．

(4) level j の user は 2^j unit の時間が割り当てられる．1 unit は，0.5 秒である．より高い priority の user が入ってくれば，先取りされる．

(5) 割り当てられた時間が過ぎても終了しないと $j+1$ の level にうつされる．

(6) level j の user が 60 秒待つと，$j-1$ の level にうつされる．

(7) 始めの level の決定は，プログラムの長さで決まる．その level で定められた時間と，そのプログラムを load する時間とが一定の比になるようにする．

(8) 高い priority のものが入ってきて，いま実行中のプログラムが，新しく入ってきたものに割り当てられたより長い時間を用いているならば，つぎの単位時刻で中断される．

(9) User が state (2) または (3) を離れるときは，queue から除かれ，その state に入るときは，新しく level が割り当てられる．

(10) その level の割り当ては，プログラムの長さできめる．過去の実行時間には無関係にしている．

(11) back grand system は，常に queue の最後にいると考える．

(12) background をある定められた割り合だけ確保することもできる．これは，定められた割り合いで priority をつけることで行なう．

Accounting の policy

(1) 各 user は，central processor を使用した時間に対して支払う．

(2) core に load や dump するのは，その user の前後の user との関係でどちらかが支払う．

(3) background system は，自分の load や dump には支払はない．

(4) それ以外の user は，自分の load time に対して支払う．あとが background である user は，background の loading に対して支払う．

（5） user は自分の dump に対して支払う．もしより高い priority のものによって dump された場合は，後者が支払う．

background を dump させるものは，やはり，後者が支払う．

Storage Algorithm

storage の allocation は，現在の MAC では非常に単純な方法を用いている．program の load と dump が計算と同時には遂行できないし，user's program の relocation も行なわず，いつも 0 番地から load している．core の中には，一人の user の program しか完全には入っておらず，core memory B は，常にその user には全部が使える状態になるが，その分だけ drum に前の user の program は dump される．

7. Operating Program

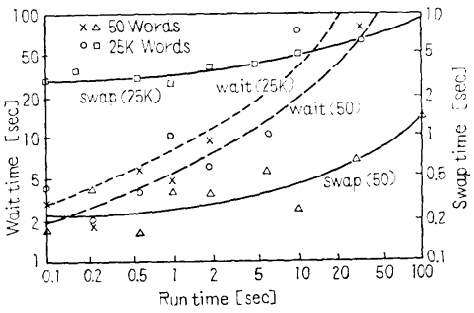
MAC System は memory が中心になった system と考えられる．その核となるものは，Supervisory Program で，さらにそれを取りまくものが System command であり，そのほかに user's file がある．Supervisor の役目は，前節に述べたとおりであり，System command は，user が System に要求を出す命令であり，その命令に対応する program が diskfile にあり，command が発せられると，その copy が作られ，load され実行されるのは，user の program と全く同じである．これらの command の中には，start, list, save といった単純なものから各種の言語の assembler や compiler に至るまで数多くあり，これらは次第に増加しつつある．現在，command は，80 種になっているとのことである．

FAP, MAD, MADTRAN (Fortran を MAD に変換する), やCOMIT, LISP, SNOBOL, ALGOL, SLIP, GPSS, COGO, STRESS, OPS, AED-0 などの problem oriented の言語も含まれている．全体のシステムの語数は，Users file を除いて，500 K 語，このシステムのために書かれたものは 50 K 語を越え，他のものは，MIT その他で開発されたものを改造して適用している．また typewriter の端末だけではなく，他の計算機，PDP-1 や 1620 との結合，特別な図形制御装置との結合，それらの Hard と Soft との各種の仕事も，このような real time on line の大形機を利用できるために可能となったものも多いと思われる．

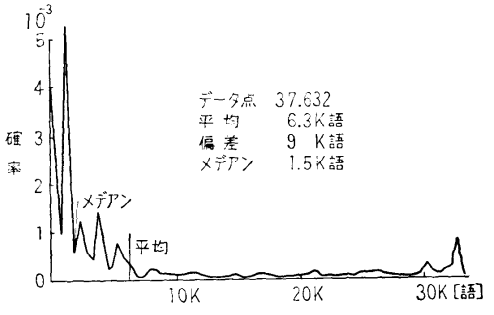
8. 実　　績

MAC System が現在の形になったのは，63 年 11 月からである．1 日 24 時間，1 週 7 日の稼動，1 日 4 時間の保守，disk dump と load, non-time sharing の仕事で，on-line の user の計算機使用時間は，64 年 4 月は，311 hr., 5 月が 297 時間，user の開始から終了までの時間の和は，これの 17 倍である．各 user の wait time と swap time とを program run time の関数とした実測値を**第 5 図**に示す．prog. の長さは，50 語と 25 K 語，user の数は，だいたい，17～20 人くらいで，各点は 30 個の測定の平均である．また**第 6 図**は約 3,700 の例からプログラムの大きさの分布を求めた結果を示す．**第 7 図**にコンソールでの人の thinking time の分布を示す（入出力の時間も含まれる）．**第 8 図**に user のためにプロセッサが仕事をした時間（Swap 時間を含まず）を示す．

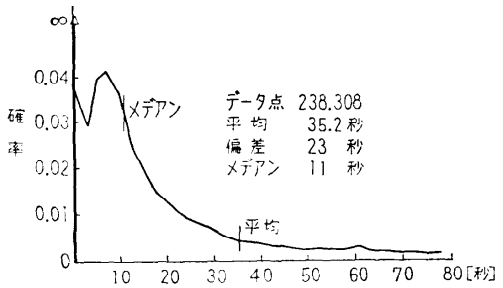
信頼性に関しては，Hardware の故障は通常の率，しかし，real time operation 特有の困難さを経験した．故障の生じたときの状態を再現することが困難であり，hardware, system program, logical design



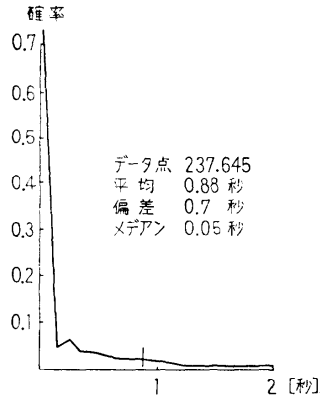
第 5 図



第 6 図



第 7 図



第 8 図

error なのかがわからなくなる．disk の protection には，最大の努力をし，1日に2回 copy をとっている．

このような仕事の結果に熱狂する人も，文句をいうのも多いとのこと，これはいつも overload でつかえなかったり，欠点や故障で使えなかったりするためである．他の utility ほどの信頼性は，まだ得られていないが，いままでの経験では，ある人達の研究開発の目的である computer system が，同時に他の人達の effective な working tool にすることが可能であることが示されたといっている．

有利なことは，private computer と同等なものになることだけでなく，大切なのはいろいろな problem-oriented language が手近に使えて仕事ができることであった．user が新しい command をつくり，このシステムに加え始めたので，ますます有力になる反面，能力が大きくなったため user にどのようなものが利用できるかということか知らせるのが困難になってきた．これには，端末を通して要求すると改訂や新しい情報が与えられるようになっている．

9. MAC システムの問題点と今後の発展

現在までの MAC は，その pioneer 的努力により大きな仕事をしてきたと思うが，現在の MAC System の非能率な点をあげてみると

（1） Command が完全な Reentrant でなく，user ごとに copy が作られる．

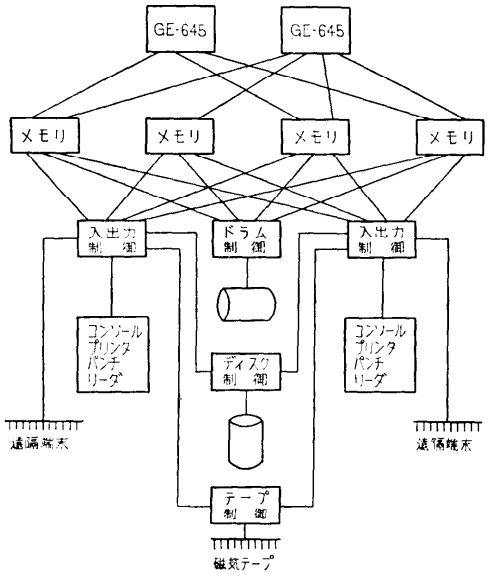
（2） 必要以上に Sub-routine の集りが，core に load される．

（3） dynamic relocation が行なわれない．multiprogramming ではない．

これらは，Hardware が従来の考え方に成り立っているためでもある．また必要なばう大な記憶を効率よく取り扱うためには，計算機中心のシステムよりはむしろメモリ中心であり，また message-store-and-forward communication の進化したものになる．また multi-processing の形で処理を行ない．しかも各機能がそのときの負荷に応じて dynamic に割り当てられることも必要となるであろう．MAC の目的である computer utility を考えるならば，システムの面でも Hardware の面でもまだ初歩の段階にあるような気がする．

Software においても，従来の計算システムに適した Software が必ずしも real time での time sharing system に適した方式とは限らないように見える．compiler だけではなく，ある場合には，interpreter の方式の方が有利な場合も十分に考えられる．これも過去にこだわらず新しい見地から検討する必要があろう．

MIT の project MAC では，いままでの経験をさらに発展させ，前述の欠点を克服して，目的とする utility の性格を目ざして，従来のシステムを大きく変えようとしている．これは Multics (Multiplexed Information and Computing Service) とも呼ばれている総合的なプログラムのシステムを完成しようとし，まず GE 645 に対して適用しようとしている．その構成は，**第9図**に示すように，4個の Memory modules に，2個の中央処理装置，入出力制御装置などが結合しており，使用者の端末は，通信線で入出力制御装置に結合している．この multi processor, multi memory などは，いずれも非同期で独立に結合されており，今後の変更，拡張に対処できるように



第 9 図

し，Supervisor も PL/I を用いて現在製作が進行中である．

GE 645 は，MIT の主張に従って，2次元アドレス方式を採用している（IBM/67 も．ほぼ同様な考え方を採用している）．これは，プログラムやデータは，セグメントからできていると考え，その大きさは任意で，使用者は，記憶装置のことを考えないで用いてよい．機械の方では，セグメントはさらにページからできていると考え，これをコアメモリの物理的な場所に割り当てるが，これは固定した場所ではなく，必要に応じて変わる．あるセグメントに含まれるページは，どのコアメモリの位置に対応するかということと，そのページの内容を変更できるかできないかといった制御の情報をもったテーブルが必要となる．

virtual なメモリから physical なメモリへの変換をいつもやっていては大変だから，ある時期に多く用いられるセグメントとページに対しては，小さな associative memory を用いて直接変換を行なう．このページ方式を上手に管理することによって，高速に動的にメモリの割り当てが可能になるだけでなく，高速のメモリを有効に使用できるようになる．たとえば，コア・メモリに要求するページがないならば，そのページを外部メモリから，コアにもちこむこと，また各種のメモリ保護が容易になる．その結果多数の使用者の要求を効率よく干渉することなしに処理できるようになる．

ソフトウェアは，ことにセグメントとページ方式に関係する所は重要である．セグメントは，名前で呼び出され必要なセグメントは動的に結合され，自動的で高速，これは他のセグメントの内容を参照するときも同じである．プログラムは，reentrant で recursive な用い方ができる．使用者は，めんどうなセグメントやページのことを知らなくてもよい．supervisor はこのほか scheduling や accounting も取り扱う．

ファイル・システムは，time sharing では重要な部分で，それの呼び出し，書き込み，秘密や保護に特別の注意が払われる．遠隔ターミナルは，主として，モデル 37 のテレタイプで改訂 ASC/11 コードを用いる．また IBM 1052 のターミナルも用いる．これらの端末は，もっとよいものが考え出される必要があり，殊に，図形の入出力装置のよいものが望まれる．このほかシステムの信頼性をあげる手段として，機械の構成を変更したり，分割したりできるようにし，新しい supervisor の開発や，保守や不良箇所の修理にも備えている．この Multics に対しては，すでにその計画のより詳細なことも発表されているが，いずれも現在開発中のものであり，成果のわかってくるのは，2～3 年先のことであろうと思われるから，それらの紹介は，次の機会にゆずる．

参考文献

1) S. Boilen, 他：A time-sharing debugging system, Proc. SJCC, 63
2) T.M. Dunu, 他：Remote Computing, Proc. SJCC, 64
3) J.C. Show：JOSS, Proc. FJCC, 64
4) J.I. Schwartz, 他：A general purpose time-sharing system, Proc. SJCC, 64
5) H.A. Kingslow：The time-sharing monitor system, Proc. FJCC, 64
6) F.J. Corbato：An experimental time-sharing system, Proc. SJCC 62
7) F.J. Corbato, 他：The compatible time-sharing system, MIT Press. 5/63, second edition (P.A. Crisman editor) 5/65
8) R.M. Fano：The MAC system. -MAC-TR-12, 10/64. この他多数の MAC 関係の報告を参照，Proc. FJCC 1965. Multics 関係の論文，および Time-sharing 関係の論文を参照した

（昭和 40 年 3 月 4 日受付）