Section I.

This design notebook is a first tentative attempt, to define an operating time-sharing system for the Project MAC modified G.E. 635 computer. Even though the notebook will usually be written in a definitive style, it should be understood that it is really meant to stimulate discussion, clarification, and improvement as well as to reveal the hidden problem areas. Throughout the design, even though it may not be realized on the present equipment, there should also be understood the long range goals as well as the contemporary compromises.

These long range goals are:

1. To operate a computer utility system on a continuous basis, 7-days a week, 365 days a year much like the phone or power companies. This definitely means that system failures must be infrequent and of short duration (e.g. few seconds).

2. The system must be able to accomodate large numbers not only of typewriter users, but of a spectrum of terminals such as scopes, plotter tables, etc.

3. The system must be able to "baby-sit" real-time experiments or equipment with definite guaranteed response times (e.g. a few microseconds) with definite levels of processor activity (e.g. a few milliseconds) with a definite duty cycle (e.g. a few seconds).

4. The system must be capable of partitioning such that
   a.) each part of the system can do preventive maintenance and checking on the other part,

b.) extra reliability can be programmed by doing duplicate or checking calculations on different processors and

c.) the processors can be applied to separate and distinct time-sharing systems. One example would be a sub-system operating the tracking of a radar antenna while another sub-system maintained normal time-sharing. Another example would be the system programmers trying out a new supervisor for an improved or different time-sharing system than the one in normal use. As much as possible, system operation and partitioning should be automatic.

5. The system programs should be as modular as possible in order to enhance clarity, maintainability and upgrading. There should be as little user context as possible in the central supervisor. (It is doubtful that this should change until there is a widely accepted universal language).

6. The system must have symmetric use of multiprocessors, effective multiprogramming such that no individual user must concern himself with concurrency and the ability to smoothly write programs which are pure procedures capable of being operated simultaneously by many users.

7. An important objective is to develop a system programming language which is truly machine-independent. In particular, the bulk of the system programming and all user programs would be expressed in this language so that the system could be transferred from one computer to another without changing more than, say, 25K to 50K of code. The transferred system would only have to run at a few

percent of the effectiveness of a subsequently "tuned" version of the system for this independence to be invaluable. In other words the user would be able to count on indefinite growth of his programming developments without periodic catastrophic destruction of all his programming tools. Present-day algebraic compilers already approach this goal but fall short in the data representations (i.e. the user is quite aware of and the programs are dependent on the word size and frequently on a few other idiosyncracies of the machine). The data-representations for machine independence can obviously be done and the only issue will be comparative efficiency with hand coding. Efficiencies of as little as 50 percent or less would still be of great use for most of the system especially if sensitive modules were done by hand coding.

8. Further goals are incremental compilers and fully symbolic debuggers such as MADBUG.

9. Finally, the entire system will be written in the form of program segments with all programs, with perhaps a few exceptions, written as pure procedures.

Many of the above ideas are already contained in the necessary background material to this notebook. Essential references are:

1. The CTSS programmer's guide and all current operating systems bulletins, etc. as revised by P. Crisman.

2. MAC Technical Report TR-3 by F. J. Corbató, giving the viewpoint of a multi-user, multi-processor, multi-programmed system.

3. MAC Technical Report TR-11 by J. Dennis giving the structure and philosophy of program segments.

4. MAC memo M-182 by E. Glaser giving a complete first-cut at the modifications to be made to the GE 635.

5. An informal note by J. Couleur of Nov. 20, 1964 giving the revision II description of the 635 modifications. (This note is somewhat incremental to M-182.)

6. CC memo 241 by R. Daley, R. Creasy and R. Graham, giving the specifications of the new disk and I/O control programs.

7. Programming Staff Note 32 by M. Bailey and R. Daley, giving the broad specifications of disk editing and maintenance.

Having roughly established the goals, and the background, it remains to discuss broadly the short range goals. It should be understood that the present CTSS system is the standard of reference and that most ideas will be discussed incrementally.

1. As a basic minimum the first version of the new system will be at least externally to the user similar to the present CTSS system. That is, these will be all the usual commands with the same names, etc. However, this system must have the following essential changes made: a) all typewriter I/O should be in terms of a single, full character set (i.e. 8-bit code); b) all programs, system or user, must be in the form of program segments; c) all new (i.e. not just recoded or borrowed) programs will be in pure procedure form and as efficiently segmented as possible; d) public libraries

and commands will be operated from the start as pure procedures; e) automatic segment-turning (and possibly automatic page-turning) will be done from the start for all programs (i.e. user or system) so that efficient multiprogramming will result; f) the initial system will handle n (where n≥2) processors as a symmetric pool; g) accurate system usage accounting will be done from the start; and finally h) monitoring will be built-in for system performance.

2. As much of the standard GE software as possible will be used in the system. In particular consideration should be given to absorbing code for the control of terminals, the disk control, the general loader, as well as adapting the assembly program, Fortran IV and Algol.

3. The CTSS programmer's guide which is currently being revised should be a direct antecedent of the new manual. Descriptions of all programs (i.e. 1-page writeups of subroutines and commands or their equivalent) should be a part of the complete manual.

4. Although the new system should be self contained, i.e. the only background is Foreground-initiated-background (FIB) with central tapes and cards controllable from the consoles, provision should be made so that GECOS will operate as an independent background. This will ensure rapid acceptance of our system by GE users who are only able to start time-sharing as a pilot operation while they persuade and educate their users as to the benefits of conversion. This compatibility however must not introduce any

serious comprimises or delays into the development of the new system.

5. The 7094 compatibility mode of the GE 635 will not be allowed under time-shared operation. (The special equipment required to do this will not be installed at MAC.)

6. The GEM assembly program will be modified to allow pure procedure and data region outputs. Until alternatives exist, the system will be programmed in this language.

7. In order to have a useable operating time-sharing system by Oct. 1965, there will only be a single time-sharing system prepared, possibly by editing over to GEM programs manually the current CTSS modules, basic commands and subroutines which are still pertinent; better languages should be used whenever available. The procedures should be pure procedures if at all possible. Up to April 1965 a 635 simulator should be available (under batch process IBSYS) on the 7094 for checkout. From April to Oct. 1965 a standard 1-processor, 2-bank memory GE 635 will be available. In Oct. 1965 a modified 2-processor, four-bank memory system will be installed.

III. Things to do in the new system

To further delineate the new system, an attempt will be made to partition the tasks into more manageable areas. A partial list of categories which are not particularly in the order of importance or difficulty are:

1. Segment structure and format convention specifications

1a. Subroutine Linkage

2. Specification of overall operating system strategy

2a. Segment and page turning management.

3. Disk file I/O control.

4. Disk file editing, maintenance and restart procedures

5. Magnetic tape, card reader, punch I/O control

6. Typewriter I/O control

7. Hi-speed line I/O control for the ESL console, the PDP-6, and the 1620.

7a. Real-time controller

8. Modifications to GEM for segment compatible output

9. Modification to GE Fortran IV for GEM output as segment compatible

10. Bootstrap loader for the system.

11. Resource allocation (i.e. sched. and storage)

12. Accounting of usage

13. Monitoring of system performance

14. Foreground initiated background

15. Interconsole messages

16. Magnetic tapes controlled from consoles

17. Macro commands

18. Off-line manuals and documentation

19. On-line documentation

20. Development of specifications for a machine independent system programming language

21. Basic commands

22. Basic subroutine library

23. Recoding the MAD translator

24. Specifications for partitioning multiple systems and for multiple processor use by a single user.

25. Specify subprogram binder (i.e. loader) for segments

26. Specify conventions of data files.

27. Debugging tools