informatics inc. 5430 VAN NUYS BOULEVARD / SHERMAN OAKS / CALIFORNIA 91401 / PHONE 783-7500

IC-68

24 September 1965

I am enclosing a copy of our report entitled, "MTACC Computer Technology Exploration." This report is a technical working paper, designated MWT-37.

Very truly yours,

INFORMATICS INC.

Irving Cohen

Irving Cohen
Vice President
Command & Control Systems

IC:jj

enclosure

Submitted to:

OFFICE OF NAVAL RESEARCH
Washington, D. C.

Contract No. NR4388(00)

Report No. TR-65-58-19

# MTACC COMPUTER TECHNOLOGY EXPLORATION

20 September 1965

INFORMATICS INC.

5430 Van Nuys Boulevard
Suite 500
Sherman Oaks, California 91401

## TABLE OF CONTENTS

TABLE OF CONTENTS (Cont.)

# LIST OF ILLUSTRATIONS

## Section 1

## INTRODUCTION

Computers have had more impact on the development of command control systems than any other technological development. They provide the ability to perform complex mathematical manipulations and handle vast quantities of data with speed and precision. Due to the increasing complexity of modern warfare, command control systems of the future will continue to rely heavily upon the ability provided by the computer. Advances in computer technology in the past have been steady and impressive.· Projected capabilities are equally impressive.

It is with this thought in mind that computer technology was explored for the period 1975 to 1985. Throughout reference is made to an MTACC system. This is the generic name of a future improved capability Marine Tactical Command and Control System. It's outlines and functional requirements are as yet not fully developed. But the advances in computer technology justify a more probing search of areas of applicability to amphibious operations than has been performed heretofore. This explanation, done in parallel with those for displays, communications, input/output equipment etc., are primarily related to amphibious operations and ground combat. This study continues where ANTACCS first year technology exploration ended and presumes that the study is a basis for many of the derived comments contained herein.

In the projected Marine Tactical Command and Control era, computers will be capable of providing the field forces with small, rugged, field transportable, battery-operated computers that ten years ago required tons of equipment and kilowatts of power. This improvement in capabilities with a reduction in size is attributable both to the use of integrated circuits and to advances in computer design and organization. The following report is concerned with the advances in computer design, organization, and utilization. The report however, does not discuss the mechanics of construction which make possible the fabrication of the small, reliable, and relatively inexpensive computers of the future.

One of the more important tasks in developing technical system concepts for MTACC is to determine what organizations of computers and computer systems are best suited for fulfilling the functional requirements. This analysis is of vital importance to a future MTACC design. Results of this analysis will be used in the Technical System Concept effort to select a best approach within the constraint of operational requirements, reliability, redundancy, commonality, and cost effectiveness.

## 1.1 REPORT ORGANIZATION

The MTACC Study Computer Technology Report was designed to fulfill a number of objectives. It describes expectations of future computer technology; it explores how these technologies may be applied in general to various areas of amphibious operations. In this sense it describes candidate areas for computer implementation. As the functional system requirements are not yet complete it is premature to state that computers should be used at different points of a future MTACC system. The many requirements of mobility, availability, maintainability and logistic support have not yet been fully probed. In the course of the MTACC study Technical System Concept effort these areas of applicability and the justifications for them will be developed.

In addition to the provision of necessary data processing capabilities for tactical operations it is necessary to explore those areas of computer technology which have a heavy bearing upon system realizability, practically speaking. That is, it is necessary to see the impact of advancing technology on the cost of acquisition of hardware and software, the modification of systems to meet changing requirements and the provision of intra and inter service compatibility. Therefore such matters of technology that make for minimal cost, retarded obsolescence and greater interface compatibility are also treated. The problems currently anticipated by other services such as uniformity and decreased expense of program production are treated.

The style of presentation is adapted to the complexity of the subject material. Where relatively involved concepts are met a more tutorial approach is taken. The purpose of this technology report is to provide a basis for further technical system concept activities which will be responsive to the developing functional system requirements. It will act as a guide document to the use of advanced data processing techniques.

The MTACC Computer Technology Report is organized into six major sections and six appendices. Section 1 is the introduction to computer technology setting forth study objectives. Section 2 is a summary of the findings of the study and includes conclusions and recommendations which are documented. Section 3 presents a description of the state of the art which exists in military computer technology. Section 4 documents MTACC computer systems considerations. These include multiprocessing, memories, organization considerations, and software trends. Section 5 lists design concepts and application characteristics of computing systems and includes a discussion on family concepts and compatibility. Section 6 discusses the implementation concepts of computers including the relationship to the CCIS-70 computers. The factors governing selection of a contractor and a system are also included. The section is concluded with a discussion of the implementation phases and principles which govern computer development.

The six appendices present more detailed information on the functions and applications of computers. Appendix A presents detailed specifications of two computers referenced in the study. Appendix B discusses general multipurpose registers and Appendix C gives a description of the main types of memories. Appendix D discusses the major computer language which have direct application of the study. Appendix E is a glossary of phrases and terms used in the report. The glossary is included as an aid in understanding technical terms. Appendix F lists the source material used in this report.

# Section 2

# SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

## 2.1 REPORT OBJECTIVES AND STUDY PURVIEW

Computers and automatic data processing will be an important part of a future MTACC System. Tactical command and control increasingly has become computer oriented. Despite certain special Marine Corps requirements, this computer orientation is likely to be true for many Marine Corps tactical operations. However, acceptance of this point of view raises many questions that must be answered. All aspects of computer technology must be evaluated with respect to impact on an MTACC System.

A review and analysis of computer technology is an important part of the MTACC study. The subject of this report is computer technology suitable for application to Marine Expeditionary Forces requirements. In the ANTACCS effort, where shipboard data handling systems were emphasized, computer technology was examined from the point of view of components and systems. The computer systems work emphasized larger scale militarized computers, but not in all cases fully ruggedized and transportable. These computers were more general purpose in nature such as those frequently used in strategic systems. This study will emphasize computer technology oriented to the particular problems of the Marine Corps.

The subject here is computer technology related to militarized, ruggedized, transportable equipments. It is desired that the computers under consideration be capable of handling all future tasks concerned with Marine amphibious operations and land combat.

The computers to be used in the future MTACC systems should be appropriate for implementation in a tactical data system which would become operational in the 1975-1985 time frame. The information is developed in a framework of current computer technology and appropriate forecasts are made to develop insight as to the desired and potential capability of the 1975-1985 period. To the extent possible at this stage of MTACC, the information is developed within the framework of the requirements and systems concepts which are evolving in the study.

More specific statement of the objectives of this study are as follows:

1) A review and analysis of the technological state-of-the-art for ruggedized, transportable computers.
2) The development of required and desired capabilities of computers for Marine Corps use.
3) An analysis of the hardware and software implementation concepts for computers.

Computer technology will be important for any future Marine Tactical Data System. This study's objective is to determine what may be available, what the Marine Corps will need and how the equipment may be selected and developed into a useful capability.

The study emphasizes the systems aspects of both hardware and software with relatively little documentation on circuits and components. The circuits and components aspects of computers were covered in the ANTACCS program and are under continuing investigation. The aspects of the technology emphasized here are those most closely associated with the user.

It is emphasized that the computer technology study up to this point is a modest effort. Although important beginning steps have been taken, and a number of important conclusions reached, more remains to be done, especially after functional system requirements and concepts are more specifically determined.

## 2.2 COMPUTERS IN A FUTURE MTACC SYSTEM

There is little doubt that computers will be important to a future MTACC system. This technology study is based on that assumption. Although it would be desirable from many points of view to häve a clear cut picture of exactly what the computers will be used for, and how. they will be used, there is much which can and should be done regarding basic technology until such detailed computer use information has been developed. It is too early in the MTACC program to state how much computing will be performed and for what purposes computers will be used. It is, however, possible to discuss system configurations in considerable detail. Perhaps, more accurately, it is possible to discuss the required characteristics of systems with regard to configurations in some detail. For example, questions of modularity and multicomputer operations are appropriately set forth as principles to be followed. Many aspects of logic design can be specified at this time. Physical characteristics and environmental requirements are reasonably well known. It is also possible at this time to develop detailed characteristics of computer modules which would be responsive to Marine Corps requirements since the overall characteristics of the real-time data processing problem are known.

On the other hand,many questions can not be answered at this time. For example, the required total speed of computers as they are aggregated in multicomputer systems cannot be determined at this point. It is also not possible to state quantitatively the requirements for: internal storage, auxiliary storage, number of input/output channels and the like.

An important point of view developed in these computer technology considerations is, that the earliest questions on computers referred to the technology itself and are only dependent on the functions and applications environment to a limited extent. The overall character of the data processing problem, the environment, and design criteria of reliability and maintainability and the like, are matched with the present capability and the extrapolation of that capability. Although the results do not yield a completely designed computer system, the broad outlines of a computer system have been developed.

Considerable work needs to be accomplished, however, to determine what configurations are best and what kind of peripheral equipments should be developed and used with this computer module..

Many questions need answers. What is the lowest command level at which computers are applicable? What is the entire spectrum of potential applications of computers? How ambitious should system designers be in the degree to which they should strive for automation? These questions, of course, can be answered by answering prior questions relating to the overall functional system requirements, and the functional system concept.

## 2.3 CONCLUSIONS AND RECOMMENDATIONS

In the following sections a number of conclusions and recommendations are summarized:

### 2.3.1 State-of-the-Art

An examination of the state-of-the-art of computers and computer concepts suitable for use in an MTACC was made. This was not a thorough or exhaustive study. However, the following conclusions were developed as a result of the analyses and studies conducted:

1) No contractor in the country today has developed a family of computers and a system concept completely suitable for an MTACC system. A number of contractors have developed prototypes of modern computers which could be points of departure for a future system. These contractors are developing the equipments with their own funds, for the most part, and are aiming the computers at broad application areas. However, much of the work done to date is directly applicable to the Marine Corps needs. There has been little accomplished, however, in terms of total system and family concepts which are sorely needed. A few contractors have developed these broader concepts to the point where they can be described in general terms. However, there is no specific design for families or systems of computers of the kind needed for an MTACC system.

2) The speed of the computer modules under development today is considered adequate on the whole. Most manufacturers are developing computers with memories in the 1-5 microsecond memory cycle range with equivalent add-type instruction times of 2 to 10 microseconds. In other words, computers are in an advanced state of development now, have 4-10 times the capability of the current ruggedized computers such as those appearing in inertial guidance systems of intercontinental ballistic missiles.

3) A majority of the contractors have under development a computer module which is suitable for airborne applications and could be extended to field combat use. Many, if not most, of these have not given much thought to modularity, multicomputers and computer family concepts. Some, however, have given some preliminary thought to these family and systems concepts although in no case are the concepts developed to a suitable extent. Questions of details are lacking although it is clear that the modular computers developed by these contractors are suitable for extension into bigger systems. These contractors have undoubtedly displayed a capability to develop the broader systems, given sufficient contractual support.

4) Computers under development today seem to meet most environmental and physical factor requirements. The extent to which these prove out in actual application and in field tests of product line manufactured computers is another matter however.

5) The implications of new circuit design to reliability have not been sufficiently treated. There is some doubt whether adequate analysis has been accomplished to determine optimum package design in view of maintainability requirements.

6) Software does not exist in any quantity for any of these systems. In few cases there is nothing more than machine type assembly language. Furthermore, there is no standard appreciation to the development of software because of the lack of unanimity of opinion of higher order languages for these systems. However, there is evidence that contractors appreciate the problem and have a capability to solve it, given the time and contractual support.

7) The incorporation of special language features and the addition of new compiler language statement for multicomputer usages will be an important consideration in developing programming facility. The language design should be flexible and should incorporate software modularity; and should be as nearly parametric to the anticipated applications as is possible.

In this connection, the development of a special command/control language is recommended. This language should include special multicomputer macros to treat precedence, scheduling, and task allocation problems. In addition, it should be sufficiently generalized to be to some extent translatable through switch action and should also have display feedback characteristics.

8) The biggest technical challenges lie in systems organization and in programming systems. A systems organization should be developed which makes the systems open ended and the programming orders must be developed to make programming more efficient.

9) There is a general lack of militarized and ruggedized peripheral equipments. A future MTACC system will require auxiliary memory and printing devices, for example. It appears that developments have not proceeded as far as they should in these areas.

### 2.3.2 Computer Technical Requirements

The following summarizes some of the salient requirements for a future MTACC system:

1) The computer systems should have as a component at least one type of general purpose computer module. This computer would have an order structure of general purpose characteristics with the need for information handling emphasized in the order code as well as arithmetic instructions. The module should have all the characteristics to meet the requirements and applications relating to time sharing, multicomputers, and the like. In addition, it would be highly desirable to allow higher speed memory to be substituted as required for lower speed ones and larger memories to replace smaller memories, depending on the application needs.

2) A system should be designed which accommodates the most modern multicomputer concepts. This will allow system units to be aggregated to allow increased capability for some applications while still using the same basic modules. A systems concept should be developed which is responsive to real time environment and time sharing and which allows the addition of peripheral equipments of a wide variety of types and in small or large numbers.

3) Special attention should be given to requirements of the real time environment and time sharing. Items such as the following should be included: modern interrupt handling, modular and high capacity input/output channels, memory protect devices. An important requirement is that compatibility and commonality be achieved to a very great extent. This applies to circuit modules, memory modules, peripheral equipment, controllers, and the like.

4) Although higher speeds will be available in the 1975 to 1985 period, some current applications indicate that a 1-3 microsecond memory cycle and a 2-6 microsecond speed for air add type instructions may be sufficient.

## 2.3.3 Family Concept

A single system computer family should be developed for, or used by, an MTACC system. Recent military and commercial developments underscore the efficacies which can accrue from integrated hardware and software approaches. Many of the requirements, which commercial users are facing and reacting to, are the same as the requirements for the military user. The objectives of low hardware and software costs and system maintainability are generally very similar.

It is visualized that the computer system can be one computer system. This system could be implemented by one general purpose computer module. This computer module could be used by itself in modest sized memory configurations for the lower echelon applications, and it could be used in multicomputer configurations for higher echelon applications. The system must be designed so that it is open-ended with regard to internal memory sizes, auxiliary memory, input/output equipments, number of modules in multicomputer configurations, and display device attachments.

It appears possible to design a system so that it can be used in configurations of equipments, or under certain field demands "pulled apart" and used in smaller configurations. It is emphasized here that what is meant is the grouping and regrouping of the equipments into various sized multicomputer and multimodule systems according to system needs. If two units are separated to meet contingencies, it should be possible for each to take a portion of the system which has been used to support the total.

Abstractly, one can look at the computer application for Marine Tactical operations as filling a matrix, the elements of which are defined by intersections of echelons and applications. The echelons are Divisions, Battalions, and the like, and the applications are fire support, intelligence and similar functions. One can then initially analyze the problem by looking at "vertical and horizontal" aggregations of the echelon-application designed matrix elements.

Software would similarly follow the family concept. There would be one basic executive program, for example, which would be used for the systems, independent of the size of the system as the various units are combined. There would be one machine language, since there is only one general purpose computer module, and it would be one higher order language for the entire computer system, or, if appropriate, one language for each major application area where each language would be usable throughout the entire system of modules.

The family concept would include coordination of compatibility and commonality to a very great extent. These objectives can be reached to a very great extent for the computer system within an MTACC system. Very likely the compatibility and commonality would be reached to a very significant extent by the MTACC system and any future sister service system as well. It is doubtful, though possible, that there can be compatibility and commonality between this system and all elements of the DOD in the time frame discussed.

## 2.3.4 Implementation Concepts

The following represents a summary of conclusions on implementing the future computer system:

1) Early decision must be made as to the relationship between the computers of the MTACC system and those of other U.S. armed forces. The decision should cover political, administrative and technical aspects. Almost everything which is said about implementation concepts is dependent upon these broad political considerations.

2) It will be important, for instance, to determine the differences in requirements between the Army and the Marine Corps for tactical computer systems. These differences should be clearly understood at an early date regardless of the degree of commonality between any future Army system and a future MTACC system. These differences should cover items such as fire support and combat philosophies.

3) The choice of a contractor to develop the computer system should include the classical ones of engineering and manufacturing capability. However, in addition, the contractor should have systems experience and should have a demonstrated ability to develop a total systems approach to the computer problem. He should have a demonstrated capability to understand the applications and have a demonstrated capability to develop this family concept described in the previous section.

4) In the past, the general procedure for procuring computers has been along the following lines: each functional area was analyzed in detail and computer requirements developed for each area. Following this, the detailed computer specifications were developed and procurement made on this basis. This hadmany serious disadvantages.

5) The disadvantages in proceeding, as described above, are: insufficient advantage was taken of contractors' in-house development, and procurement costs were high. The result was that a basically different computer was developed for each application. Furthermore, as requirements changed as the application became better understood, the computer developed no longer fit the requirements.

6) The implementation scheme advanced here would proceed along the following lines: in general, the development of the computer system would develop along with the development of the detailed requirements. The computer system developed would be flexible in its applications and open-ended with respect to the system configurations possible. Modifications to the computer system could be accomplished as the requirements become better established.

7) In the early phases of the work two manufacturers could be selected so that each could provide a paper design and the best one selected from this competition. Since the military would be placing many "eggs in one basket" with the selection of one contractor, contractor teams cutting across rival computer manufacturers would

be encouraged. In other words, the ideas of contractor consortium and associate contractors such as those for larger airplane and weapon systems would be encouraged for these developments.

## Section 3

## STATE OF THE ART

### 3.1 GENERAL

### 3.1.1 Study Parameters

Military requirements in the field environment have led to the development of computers especially designed for this use. These computers must operate in severe and hostile environments, and must therefore exhibit high reliability, low maintenance, high maintainability, and minimum physical size.

A survey of several militarized computers currently being developed has been conducted to determine computer characteristics of the next generation of ground mobile computers. This survey was of limited scope and included a study of approximately a dozen computer systems. A detailed analysis was performed on the systems aspects of two computers which are representative of current industry developmental attitudes.

Included in this study are several computers which are, for economic reasons, currently considered primarily for avionics and space applications. These computers are of interest because of their miniaturized production techniques. These techniques allow production of large and medium scale (by today's standards) miniaturized computers which are competitive with conventional size machines. (What is regarded as a conventional size insofar as physical size is also becoming radically reduced.)

There is a growing trend for commercial and military computers to be very similar. Heretofore, this trend has come about in two ways:

1) Manufacturers of military computers have been able to find commercial applications for their machines and have marketed them in modified versions.
2) Commercial manufacturers have militarized their commercial products in the hope of attracting military business.

There now appears to be a growing opinion among manufacturers that the computing capability of a general purpose design encompasses the application span of most military requirements. This is sufficiently true to suggest significant economic advantage for using the support and development investment of commercial models, especially of their software.

The impetus for the development of what are now considered large and medium scale computers in miniaturized versions comes not only from the military applications for the ground mobile environment, but also from avionics and space applications. The basic limitation, as far as miniaturization is concerned, is not technological development; rather it is due to human and mechanical limitations (e.g., control panels, manual input devices, and display).

It is therefore concluded that computing power and the degree of miniaturization for computers will be largely sufficient for MTACC for the 1975-1985 time period. Many instances could be cited where it is important to introduce equipment whose characteristics and capabilities appear beyond the cost versus performance curve; where it may be important to purchase twice the capability at ten times the cost, as in cases of military necessity for example. However, computers do not appear in general to be in this category as far as the MTACC applications are concerned.

### 3.1.2 Study Contents

Appendix A lists detailed specifications for two computers currently in the process of final development. These Litton and Autonetics computers are realizable in the field in an off-the-shelf basis by 1970. The analysis of machines of this type presents a basis for extrapolation to the MTACC time period. The following paragraphs in this section discuss the capabilities of the two representative computers and lists the desired instruction repertoire characteristics for the MTACC system. Following this material is a description of the projected capability in computer technology.

### 3.1.3 Definitions

To provide a basis for discussion, it is helpful to review a number of computer organizational concepts. The organization of a computer is typically composed of five basic functional elements:

1) Input — that portion devoted to bringing data in from external sources
2) Output — that portion devoted to sending data out of the computer
3) Arithmetic — that portion devoted to the logical manipulation of data
4) Memory — that portion devoted to storing data and programs
5) Control — that portion devoted to control of the flow of data among 1), 2), 3), and 4).

Often the above elements share common hardware and will vary in the degree of interrelationships from one organization to another. Functionally, however, they are distinct and can be considered independently when discussing organization. Varying the capabilities of any or all of the functional elements is possible, which results in computer organizations of widely divergent capabilities.

The most common division of computer functional elements is to consider the arithmetic and control unit as one basic module. This will be referred to as the central processing unit (CPU), or simply as a processor*.

A processor may also have associated input/output capability. Thus the term "processor" is sometimes used to mean a "computer without memory."

*Unfortunately, this term may be confused with any number of other types of processors, including computer program processors (e.g., language processor, I/O processor, etc.).

## 3.2 INSTRUCTION REPERTOIRES

The list of the basic operations which may be specified by the programmer for the computer to execute is referred to as the instruction repertoire. The programmer who uses the basic commands of the computer is said to be programming in machine language. An important trend, however, is the use of compiler languages which allow the programmer to present his problem to the computer by means of a compiler program. The compiler program translates his problem statement to machine language. This method of programming is generally easier, since it does not require the programmer to possess an intimate knowledge (perhaps none at all) of the individual computer instruction characteristics.

The instruction repertoire may be optimized according to various criteria. A discussion of several of these follow:

### 3.2.1 Programming Ease

Since one of the highest cost items in a computer system is programming, this is an important criteria. Two factors in the MTACC environment trend are to make it less crucial, however:

1) The trend toward use of compiler languages tends to insulate the programmer somewhat from machine language concerns.

2) The bulk of operations programs will be preprogrammed before use in the field, and will be incorporated as a fixed part of the computing system. This suggests that a greater emphasis should be placed on performance rather than programming ease. This emphasis would be reversed if the primary activity of the computer involved the running of non-recurring job programs, as would be expected at a commercial computer installation. when?

### 3.2.2 Performance

The design of an instruction repertoire may be optimized to permit maximum performance of the object program. Efforts to achieve maximum performance with a relatively small number of basic operations have featured the development of stored logic microprogrammed computers. These computers tend, however, to be more difficult to program.

## 3.2.3 Compatibility Considerations

Machine language compatibility is an important advantage in preserving software investments. Whether standards for instruction repertoires exist in this time period is not clear. Although there is a distinct possibility that a large degree of industry-wide compatibility will exist with regard to instruction repertoires, the more important level of compatibility will exist on the compiler language level.

## 3.2.4 Cost

Cost is an important factor for all decisions. However, in the context of the overall computer system, the internal logic is not a large cost item compared with memory and peripheral considerations.

## 3.3 INSTRUCTION MIXES

Variations in the method of counting instructions may tend to obscure the true nature of the instruction repertoire mix. A few examples are given to follow to illustrate this difficulty.

For some computers the instructions, Branch less than, Branch greater than, Branch if equal, or Branch if not equal, would be counted as four instructions. For others, a single branch instruction with a conditional modifier specified in the instruction word would provide the same functions.

Comparative analyses of order codes among candidate computers are difficult to evaluate and are easily subject to misinterpretation. The number of instructions is a poor indication of computer instruction repertoire effectiveness since the method of counting instructions is variable. The use of modifiers in the instruction word lend confusion to what an instruction is and what is simply a variation of an instruction. The theoretical maximum number of instructions is $2^n$, where n is the number of bits in the instruction repetoire field. However, if variations of addressing branch options, shift parameters and specifiers of working registers are also included, n may be extended to include the entire instruction field. In this way, extravagant claims of thousands of instructions (combinations) arise.

### 3.3.1 System/360 Example

The IBM System/360, which will soon become the standard of the industry, has an instruction repertoire of only four I/O "instructions." However, in addition to this, channel logic provides six additional "commands." The I/O devices, in turn, react to control communication information called "orders." These orders are I/O device peculiar and are highly diverse. Thus in the area of I/O, a hierarchy composed of instructions, commands, and orders constitute a highly structured set of control which is hardly reflected by the literal numeration of instruction types.

## 3.4 GENERAL MULTIPURPOSE REGISTERS

Factors which effect the machine efficiency are highly dependent on the organization of the internal registers. The nature, number and organization of accumulators, program counters, index registers, addressing registers, and input-output registers determine to a large degree the instruction repertoire structure. In the area of machine organization, the use of general multipurpose registers is noteworthy. A description of multipurpose registers is contained in Appendix B.

## 3.5 CHARACTERISTICS OF TACTICAL COMMAND AND CONTROL INSTRUCTION REPERTOIRES (ORDER CODES)

The computation characteristics of tactical command control encompasses a large span of applications which correspond to typical scientific and business data processing computing. The typical repertoire of the general purpose computer is largely adequate. Several computer manufacturers make no distinctions between commercial and military computer instruction codes. This approach provides compatibility with previously written programs. Therefore, existing software inventories may be used. However, those manufacturers who design special repertoires provide a few powerful commands which give special recognition to applications.

For the tactical command control problem, the computing task requirement should be categorized according to broad functional characteristic categories, rather than by the usual categories of fire support, logistics, personnel, operations, etc. The following are noteworthy characteristics of the computing requirements:

Date Management
Information retrieval
Message handling
I/O and interrupt oriented executive functions

For this type of processing the instruction repertoire should feature the following:

Bit, byte, variable field manipulation
Table searching capabilities
Flexible addressing (e.g., literal, direct, indirect, content)
Format conversion commands
A strong set of logical commands
Provision for efficient interrupt handling

## 3.6 INTERRUPT PROCESSING

An important trend in real-time systems is the use of interrupt methods for data input. A characteristic of such systems is the requirement that input/output demands be serviced rapidly. In particular, a sensitive and timely response to command/control information input by human or electronic intervention is needed. In some cases, the computers must respond to reconfiguration commands and alter the priority of operations in real-time.

For MTACC, the interrupt technique will be very important. The importance of servicing external devices before possible loss of data suggests that interrupt control assumes a prominent position in processing priority. Viewed in this way, the aggregate of interrupt processors constitutes a high (perhaps preeminent) level of executive control. By means of the interrupt technique, events external to the computer are registered in the computer program and the computer is able to respond to new situations in a predetermined and appropriate manner.

The interrupt feature consists of the ability to impose a hardware signal into the computing sequence in order to initiate a required action. The interrupt signal will ordinarily cause a branching of control to an interrupt routine. After the interrupt routine is completed, control is usually reverted to the sequence which was suspended.

Typically, the interrupt signals that data is ready for input into computer memory. It may also, however, simply signal an external event. Another use of the interrupt is to signal the end of a previously initiated data transfer. In some cases, this notification of a completed transfer triggers a new transfer. The interrupt is also used to indicate malfunctions (e.g., power failure, parity error, etc.) Interrupt signals should be identifiable, both as to the interrupt source (which device), and as to the reason for the interrupt.

In a multicomputer system, the ability for one processor to interrupt the other processor(s) is a desirable, if not necessary, characteristic. One of the most significant tasks involves the updating of common data bases to maintain currency. An important consideration in this connection is the extent to which data input is to be duplicated.

Interrupts will ordinarily cause interrupted routines (that is, the routine that was operating is interrupted). This may result in a re-entrant condition. If the routine is re-entered prior to completion, previously computed interim results (usually in temporary storage) may be destroyed. This problem has been solved in a number of ways, usually by the individual user programmer. However, standard re-entrant procedures should be adopted and an easily specified method provided as a function of system support.

When various equipments are competing for attention, problems of priority results. These priorities relate not only to I/O requirements but to the computations presently underway. In some systems, interrupt signals may be selectively enabled or inhibited under program control. This feature provides flexibility in providing for an operational sequence based on a priority logic which may be either preset or dynamically alterable.

## 3.7 PROJECTED CAPABILITY

The advances in computer technology have been steady and impressive, exceeding expectations and fanciful predictions. However, as one surveys the technical literature circa 1950-1960, he must conclude that many of the exciting developments which were "just around the corner," still are. It is instructive to notice which areas of development have exceeded expectations and those which have proved disappointing.

The most impressive gains have been made in the area of increased speed, and we are currently in the stage of computer development where the convenient unit of measurement is shifting from microsecond to nanosecond. Impressive also, have been the trend curves reflecting a sharply decreasing cost per computation.

Virtually all components of computing systems have shared in these trends; however, the rates of improvement are not uniform. For example, speed advantages in the last 20 years range from one order of magnitude to six orders of magnitude in the following categories:

[illegible] = 6   [illegible]   [illegible]

1) Internal computer logic (6 orders of magnitude) [illegible] 12KC to 120KC = 1
2) Serial memory devices (5 orders of magnitude)
3) Printers (1 and 2 orders of magnitude) 75 lpm to 750 lpm = 1 [illegible] = 2

In general, input/output and internal communications has not kept pace with the advances in internal processing capability. Electromechanical input/output devices are not advancing in speed or decreasing in size as fast as memory and internal logic elements. Other limitations include interconnection techniques and communications capability. Inasmuch as no imminent breakthrough in these areas is likely to reverse this trend, it must be concluded that the MTACC requirements in 1975-1985 will not be "computer-limited." It is evident that the challenge of future improvement for computer systems is not in the area of increased speed, but rather in the area of computer organization, computer system organization, and the development of effective methods of channeling programming efforts. System organization innovations may have a profound effect

what about [illegible] ?

on future systems capability. These remarks are not intended to belittle the contribution of increased speed, since many instances can be cited where, in a stroke, the doubling of a computer's speed has obviated the need for painstaking programming efforts at optimization. However, the projection of speed improvements and cost reductions may be made with assurance, and extrapolated with more exactness than can predictions for greatly improved system organization or breakthroughs in programming techniques.

In the area of hardware capabilities, in particular, in the areas of internal machine logic components and memories, a straightforward extrapolation of current trends is likely to yield a meaningful projection. The base of extrapolation can be used as a guide to indicate which areas of development should receive the greatest emphasis. Viewed from this standpoint, the important topics of computer technology for MTACC are those dealing with organizational aspects of computers and computer systems, and their effective use rather than those which relate to what might be termed the raw computing power (which will almost certainly be adequate for the anticipated application).

A concensus of current predictions with respect to computer capabilities indicate the likelihood of an order of magnitude increase in speed in computer speeds, and an order of magnitude decrease in size and cost. Assuming that this trend is correct, a machine with the following characteristics could be postulated for the 1975-1985 ground mobile environment as shown in Table 3-1.

Table 3-1

1975-1985 PROJECTED COMPUTER CHARACTERISTICS

| | | |
|---|---|---|
| <u>Physical Characteristics</u> | | |
| Volume | 1 cu. ft. | |
| Weight | 30 lb. | |
| Power Consumption | 100 w | |
| Packaging | Field Case with Integrated Control Panel | |
| <u>Word Length</u> | Variable Multiples of 8 bits | |
| <u>Speed</u> | Cycle | 100 nsec |
| | Add | 200 nsec |
| | Multiply | 2 μsec |
| | Transfer | 100 nsec |
| <u>Memory</u> | Internal — 2 modules of 8192 words each | |
| | <u>Other Memory Modules</u> | |
| | Scratchpad — 256 words of 32 bit mem. 20 nanoseconds | |
| | Associative Memory Module — 4096 32 bit words, 16 search criteria<br>50 nanosecond cycle, 2 μsec search | |
| | Read-Only Memory — 2048 word modules 10 nanosecond | |
| <u>Addressing Modes</u> | Direct, Indirect, Literal, Content | |

# Section 4

# MTACC COMPUTER SYSTEMS CONSIDERATIONS

## 4.1 MULTIPROCESSING

Multiprocessors and multicomputer systems, because of their inherent redundancy and flexibility, offer many advantages for tactical command control systems such as MTACC, particularly at the high command levels. Among the most important of these that relate to MTACC design are:

1) Reliability
2) Availability
3) Survivability
4) Evolutionary system growth
5) Independent tasks can be factored or processed separately
6) Simplified maintenance
7) Reduced costs

The diverse nature of command control systems causes them to be quite amenable to use of such systems. The changing pattern of the data processing load related to the threat or the military situation is especially suitable for multicomputer approaches.

MTACC design, it is anticipated, will be heavily influenced by multicomputing considerations. The most important of these is the reliability which can be achieved through the module redundancy rather than total system redundancy. Improved performance is obtained by the automatic scheduling and sharing of peripheral equipment among the processing units. A significant advantage is obtained in cases of command post displacements in a multicomputer design, since it is then possible to divide the processing capability according to tactical exigencies.

The use of multicomputers is consistent with a modular computer organization which would also feature input/output modularity and modular memories. The salient advantage of a modular computer organization is the capability for system expansion or contraction to meet requirements of different sized applications for various size task forces.

## 4.1.2 Multicomputers, Multiprocessing, and Multiprogramming

Recent developments in computer organizations which feature parallelism in various forms have given rise to a number of terms which are easily confused. A difference exists between computers and processors which determines the difference between multicomputers and multiprocessors. Thus, a multicomputing system contains two or more computers, multiprocessing indicates the use of two or more processors in the same system. The terms are often used interchangeably. One distinction is that a multiprocessing system features processors which have one or more shared memory. The reasoning for this distinction is somewhat obscure but may be traced to the fact that a processor without any memory would be largely useless. If it has only local (private) memory, then it might more properly be referred to as a computer. Therefore, the rationale for the separate term "multiprocessing" is based on the need to describe those systems which have shared memory.*

An implied characteristic of multicomputer systems and multiprocessing is that the computing elements are capable of parallel operation upon a shared task. Thus, simply duplexing for reliability (as in SAGE) and computers operating on separate tasks (even though co-located) do not qualify under the definition.

*It is recognized that the definitions presented are arbitrary. Another usage of the term "multiprocessing" is for systems which use multicomputers and use multiprogramming techniques for the solution of common tasks.

Multiprogramming is the process of using one computer for different processing tasks on a time-shared basis. Time sharing is a special case of multiprogramming wherein the several jobs being run concurrently are originated by different users. It is getting a great deal of attention, especially among scientific users who wish to have many problem solvers using the machine simultaneously. Although there is no requirement that multiprocessing and multiprogramming exist in the same system (nor is one a subset of the other), they are often utilized in the same system to good effect. The advantages of multiprogramming are enhanced in a multiprocessor system because of the parallel processing capability afforded.

### 4.1.3 Load Sharing

The problem of optimum allocation of tasks to processors, and optimum sequencing of tasks within individual processors is greatly simplified, if the length of computer tasks may be clearly anticipated. Unfortunately, this is usually not the case. Tasks are not always divided between processors so as to equalize the load.

Even in cases in which tasks are of fixed and predictable duration, the occurrence of external events, (e.g., input request interrupts, receipt of command information, etc.) may cause computing load imbalance, and possibly enforced idle time. This problem may become particularly serious if the output of one processor is required by another processor for performance of a subsequent task.

In systems for which the processors interrelate to a high degree in the performance of common tasks, intermediate results are often passed between the processors. Frequently, one machine must be idle while waiting for needed data.

One approach to this problem involves periodic monitoring of the program progress in each processor. For this scheme, processors signal each other reporting status information, and make periodic decisions whether to cause a shift of functions from one computer to another. The logic for such decisions is not necessarily trivial and that such a monitoring scheme, if made too complex, becomes so cumbersome as to negate the advantages of load redistribution. In general, the pre-empting of one processor from continuation of its assigned tasks to accommodate higher priority tasks should be minimized.

A preferable treatment of load sharing may be obtained with the use of common function lists. With this approach, the processors are able to schedule themselves by selecting tasks from a common task list located in each computer or in a shared memory. If a shared memory is not used, a requirement is imposed that the computers notify each other upon the selection of each new task to avoid duplicate assignment on an availability basis. This scheme introduces a self-organizing characteristic to the system, but is subject to precedence limitations.

4.1.3.1 The D-825 System Example. One example of load sharing is the Burroughs D-825 system. The D-825 is a military system oriented toward the command/control problem. Its modular design allows a building block approach to applications design.

All memory is totally shared between all processors in this system; therefore, programs or program segments may be shifted about from processor to processor with ease. Also, since programs need not be associated with any particular processor in logical step of thinking of programs controlling processors instead of processors executing programs was made. The result of this concept is an executive routine that is executed by each processor when its services are needed for obtaining a new job. This program is called Automatic Operating and Scheduling Program (AOSP).

The AOSP maintains a job file for all programs currently in the system. When a processor seeks a new assignment, it runs the AOSP and if it is assigned to the program, it transfers the program from the main memory to its own memory. If the program is interrupted for any reason, the processor transfers it back to the main memory suitably revised to take account of the work done to that time by the processor.

The AOSP may also divide a single program between several processors since program branches may be specified by programmers. In this way, simultaneous parallel solution of the separate segments can be achieved.

The executive routine used with this multi-computing system was developed to provide an automatic control framework for efficiently and effectively running multi-path, parallel, real-time programs. When a processor seeks a new assignment because it has completed a program or has been interrupted, it runs the executive routine. If it is assigned to that program it transfers the image of the program from the main memory to its own thin film registers. If this program is again interrupted the processor transfers the program, revised to the new status, back to the executive routine to be drawn out again as another processor becomes available. Thus, programs or program segments are shifted about from processor to processor to permit direct response to the various interrupts in the system. Programs are never associated directly with any individual processor. Moreover, processors need not even be aware that they are picking up a partially completed job.

### 4.1.4 Programming

Multi-computer systems are probably not significantly more difficult to program than other systems. The primary complexities imposed by reason of the multi-computing aspects of the system are those of scheduling and load sharing as discussed in the previous section. To a large extent, executive control for any system deals mainly with considerations which are independent of whether the system is a multi-computer system or not. Executive control in a multi-computer system involves different kinds of complexities (not necessarily more) than for a single unit processing system.

Design of control programs for multi-computers is a pre-eminent consideration, nonetheless, and much attention should be devoted to this problem. The preparation of the executive programs is a one time expense which pays dividends in the system support afforded. The control programs provide the framework within which the operations programs are executed. If properly designed, the system user programmer may be relieved of much of the concern about the details of the system operation. In general, the user programmer should not have to be unduly aware of the multi-computer aspect of the system. The programmer should be provided with the necessary tools to specify parallelism without the necessity of being concerned with assignment of individual processors to the tasks.

## 4.1.5 Executive Control for Multicomputer Systems

Executive control philosophy has been described as a recognition of the system's tasks in order of system priority; distributing them among the system's computers; and then controlling them in the individual computers of the system by an executive routine within each computer. The design may become more difficult. To the extent that task allocation between computers, coordination and timing synchronization considerations add to the executive overhead.

There are several kinds of executive control, and within each executive program is usually a hierarchy of control to provide supervisory, scheduling, monitoring, and peripheral control. In addition, as was pointed out earlier, the aggregate of external interrupt signals together with the awaiting interrupt routine logic constitutes another form of executive control which is able to superimpose environmental conditions and occurrences into the system operation.

Several basic concepts of executive control are characteristic in current multi-computer systems. Some of these are:

1) Master-Slave Control - This is typical in systems with centralized control. This has the advantage of flexibility in the assignment to the various subprocessors.

2) Reciprocal Control - This type of control prevails in systems wherein both (or several) computers all have equal control capability, but at any one time only one processor is in charge.

3) Autonomous Control - This type of control may be appropriate for systems which divide tasks largely by function (e.g., one for I/O and one for processing). The executive programs may be relatively independent and interrelate only occasionally.

4) Shared Control - The shared task concept was discussed in a previous section. Here, as in the reciprocal control concept, the processors have equal executive control capability; but no one computer is in absolute control, and the executive concept is one of self-servicing in respect to task selection.

There has been relatively little done in terms of formal or rigorous investigation of program control of multi-computer systems. This subject is a challenging one, deserving much design consideration; preferably, early in the development of the total system design.

Unfortunately, determination of executive philosophy is usually derived after other system elements are determined and is, therefore, largely a pragmatic consideration, an after-the-fact recognition of the system structure.

The full and complete optimization of all units of a multi-computer system is a multi-queue and scheduling problem of great proportions. However, much can be implemented by the executive control program to improve the overall efficiency of the system. Although the executive control program cannot yet be expected to optimize the system according to a complex mathematical formula, it can nevertheless accomplish considerable system optimization. It can also provide the very first and important step in the research and development process which must take place to allow full use of multi-computer systems.

## 4.2 MEMORIES

Memory organization and utilization is one of the most significant considerations in the design of computer systems. It is a particularly important aspect of MTACC computer design due to the wide range of memory requirements anticipated for the system. The internal memory of a typical stored program computer performs in the following roles:

1) program storage
2) data storage
3) processing and control registers
4) calculational scratch pad

[illegible]

Main memory capacity is related to the size of all programs which must be executed cyclicly or upon demand, and to the amount of data which must be held in random access, ready reference form.

An important aspect of internal memory is its modularity, that is, the ease of adding increments of memory. Modules with capacities as low as 4000 words permit ease of tailoring memory to application requirements. There is an important trend toward use of shared memories between processors. The shared memory eliminates the need for extensive transfer of data between computers in a multicomputer system. It is important to notice that a large memory may be physically one and logically many, or vice-versa, depending on the system design. An important advantage of physically distinct memory modules is that they can be separately accessed by different processors and/or I/O units simultaneously, thus increasing the effective speed of operation. One method of using this capability with a single processor calls for the storage of instructions (operators) in one memory module and data (operands) in another memory module. Access of data and instructions may then occur simultaneously, resulting in effective speed increases. This technique, called memory overlapping, is only effective for certain applications, however.

7

The types and capabilities of memories for the MTACC era will be extremely diverse. The organization and use of memories with various speed characteristics and special capabilities present the system planner with many degrees of freedom. He must not only be aware of the various kinds of memory at his disposal, but must be alert to the special capabilities afforded by the combinative properties of hierarchies of memories.

The trend toward use of multiple memories of various speeds and capabilities may be expected to continue. One could anticipate a system design encompassing a wide variety of such memory types, each employed according to the special capacities afforded. Appendix C presents a discussion of memory hierarchies. Also included are the special topics of associative memories, read-only memories, and scratchpad memories.

## 4.3 ORGANIZATION CONSIDERATIONS

The following subsections discuss modularity considerations for MTACC and introduce the "Family of Computers" concept.

### 4.3.1 Modularity

The concept of modularity is strongly interrelated with the subject of multiprocessing. Modularity is offered as the basic rationale for the use of multiprocessing and it is therefore important to examine the various kinds and levels of modularity.

4.3.1.1 Modularity Advantages. The development of standardized modules of processing capability, memory capacity, and I/O capability provides a basis for a modularly expandable system. In general, the problem is one of assembling a mixture of computer capabilities and capacities suitable for a wide range of problem types and/or a wide range of users. If the system is constructed from a common set of physical components, a number of important advantages are realized:

1) The foremost advantage of modularity is to provide reliability and assure availability. The system MTBF is greatly increased by modularity and the concept of graceful degradation is provided.
2) The individual systems constructed from the common components may be optimized for the application, or the user. The compromises ordinarily required to accommodate a large span of applications or users may be alleviated.
3) The capability for changing the computing capability in the field is provided. This capability is important for mobility and in case of displacement where both expansion, contraction, or division of computing capability may be necessary.
4) The development of a relatively small number of modules which are interchangeable has an important effect in reducing the necessity for stocking large varieties of equipments.

5) System expansion is accommodated by an open-ended modular design permitting unlimited growth capability.

6) Economies are realized in the areas of maintenance and training.

7) A modular approach is consistent with tactical deployment concepts since it permits module reallocation, possibly within minutes.

The use of modular components has important software ramifications. A modular software design is also required and the program must be adaptable, and able to recognize increases or decreases in I/O equipment and memory capability and react accordingly.

4.3.1.2 Levels of Modularity. Various levels of equipment modularity are currently recognized in tactical systems, ranging from the component level for repair replacement to small replaceable packages to functional subassemblies and subsystems. It is expected that for the MTACC era these levels of equipments modularity will tend to merge into fewer distinct levels at a higher level of equipment package. In the field, repair will be minimized because of cost considerations due to the increasing complexity and miniaturization of equipment, the lowest level of modularity will thus tend toward the functional and subassembly level. For computer systems, the modules for replacement could thus consist of processors, memory subsystems, I/O units, and special purpose modules, such as associative memories.

The amount of stratification into a hierarchy of modularity levels, the degree of standardization of system building blocks and online interchangeability depends most heavily on the interface characteristics and whether the designs allow for plug-in connectors. The relatively low reliability of plug-in connectors suggests that their use be reserved for use between rather large units. These units would tend to be the smallest replaceable modules (replaceable by unskilled personnel).

The use of common plug-in elements is important in the MTACC environment because it permits a reduction in the number of spares and specialized test equipment which must be carried for a given complement of equipment and eases associated logistics problems. Additionally, it simplifies the training of maintenance personnel and the actual performance of the maintenance function.

4.3.1.3 System Reconfiguration. The foregoing sections have discussed the levels of modularity and suggested that one of the motivations for modularity is to provide the capability to reconfigure and to expand, contract, or divide the processing capacity. An important consideration regarding reorganizational capability is the incorporation of alternate paths to peripheral equipment.

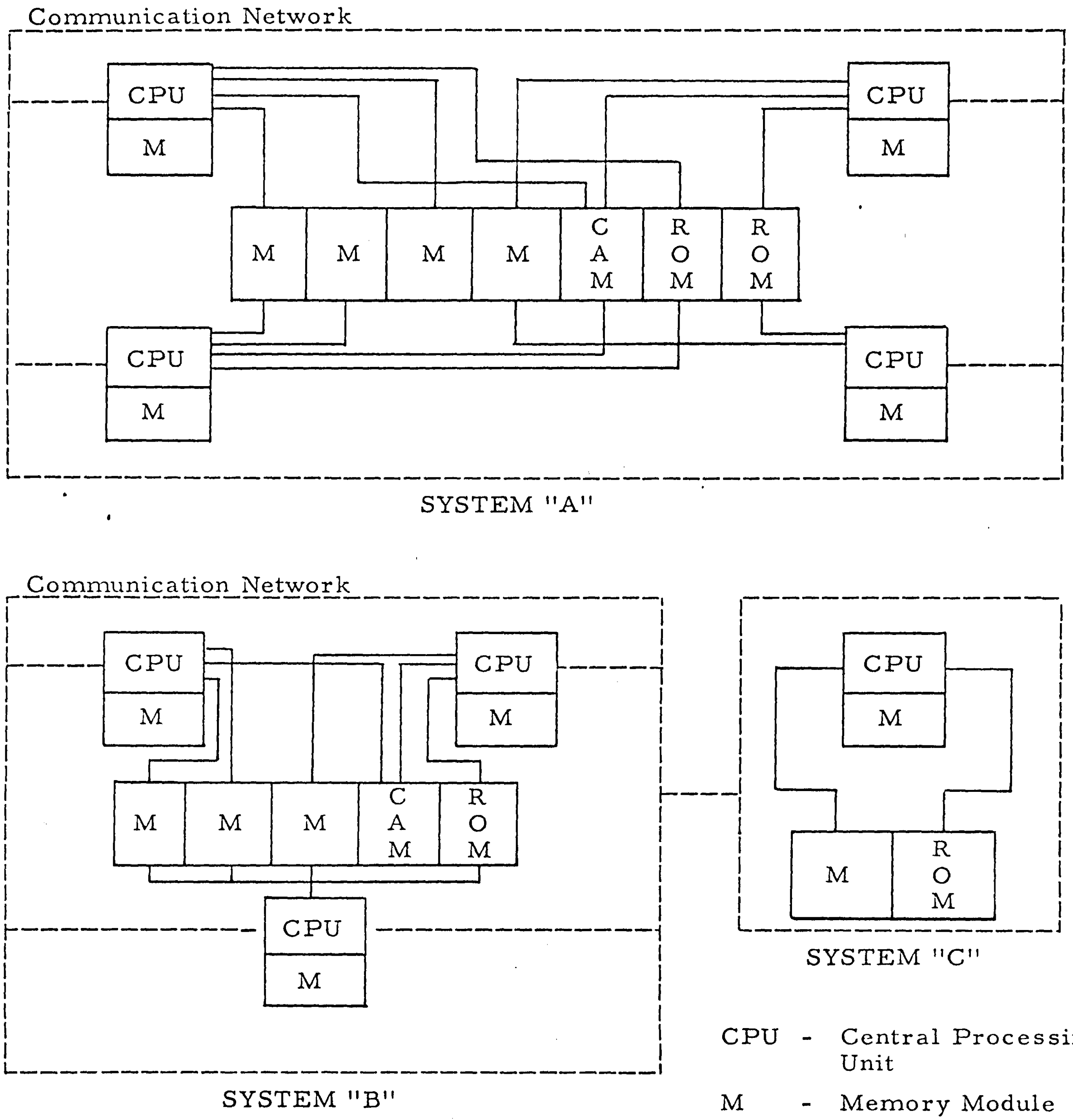
An illustration of this capability is presented in Figure 4-1. In this diagram, the basic modules are:

1) A basic computer module, consisting of a central processing unit (CPU) a single memory bank.
2) Basic memory modules (M) which may be shared between the computers and are selectable (and deselectable) either by program control or external switch.
3) Special modules which are here limited to associative memories, and content-addressable memories, but which could include other specialized capabilities as well.

This diagram illustrates the division of a computer system, (A), into two constituent systems, (B and C), either of which, for tactical reasons, could be transplaced.

## 4.3.2 The "Family of Computer" Concepts

Computer manufacturers and users frequently refer to a particular group of computers as a "family." The following discussion defines the family concept and examines the benefits accruing from the family relationships. It also considers the relevance of these benefits to MTACC.



| | | |
|---|---|---|
| CPU | - | Central Processing Unit |
| M | - | Memory Module |
| CAM | - | Content Addressable Memory |
| ROM | - | Read-Only Memory (Program Storage |

Figure 4-1. Illustration of Division of System A into Systems B and C

The "family of computers" idea has been proposed as a promising approach in military applications but has been implemented imperfectly heretofore. The concept has considerable appeal among commercial manufacturers, since it provides a basis for coverage of a complete product line.

They are then able to offer a range of operating speeds and capabilities and provide user options for a variety of systems applications.

The family of computers approach is an important design philosophy. As background for this concept, the following sections list family properties and contrast the family concept and the "unit computer" concept.

4.3.2.1 Family Properties. The family designation is usually reserved for the case where two or more computers of different size or capability are offered by a manufacturer or applied by the same user. Although successive generations of one computer may be so described, typically, the members of the family co-exist and may be chosen on the basis of their suitability for selected tasks. It is the exhibition of similar characteristics by the different sized members which distinguishes the family concept.

An important feature of new computer families is the close interrelation between the equipments (hardware) and the operating systems programs (software). Control programs are designed to complement the particular equipment configuration for each system. The supervisory programs and the equipment appear to the user programmer as an integrated structure. Compatibility, therefore, is maintained at the user level, although mechanizations of the individual computer may differ widely.

Although this approach offers many advantages for the typical computer installations, a wide selection of computing family members for MTACC is not indicated.

4.3.2.1.1 Program Compatibility. A valuable family characteristic is that of program compatibility, that is, the ability to run programs written for one member on another member without modification. Program compatibility is important because it:

1) simplifies the training of programmers,
2) reduces program development costs, and
3) permits system growth without the necessity to replace correctly operating programs.

If both upwards and downwards compatibility are not achieved in the design of the family, the third factor requires only upwards compatibility while the second requires both. Although all programs could be written for the smallest member of the family and take advantage of upwards compatibility, such an approach would not realize the full capability of the larger members. Effective program compatibility can be achieved with varying degrees of efficiency and desirability at the following levels:

1) symbolic language (compiler level)
2) machine language (instruction level)
3) by microprogrammed sequences.

4.3.2.1.2 Symbolic Language. In the first level, compatibility results from the specification of a common symbolic language for use in all program preparation, regardless of which computer may actually run the program. The individual computers of the family need not be identical at the machine language level but each must be equipped with a compiler program to generate its own machine language version of the symbolically stated program. Although compilers traditionally produce less efficient object programs than manual coding, the advantages of a common, application oriented, symbolic language may dictate its use in any case. Programming compatibility at a symbolic level would provide the three benefits, requiring that only systems programmers and maintenance personnel be familiar with the various machine language instruction repertoires.

4.3.2.1.3 Machine Language. Compatibility at the machine language or instruction level implies a strong logical similarity between members of the family. Logical functions may, however, be implemented with different electronic "building blocks" and instruction execution times may vary from member to member. If upwards and downwards compatibility is required, there must be a one to one correspondence between instruction set features. Great care must be exercised during machine design to ensure that differences in implementation do not introduce subtle (but disastrous) differences in instruction effect. Where upwards compatibility accomplishes the desired purposes, the instruction repertoires of the smaller members may be subsets of the progressively larger members.

Complete instruction set compatibility provides the three benefits stated above, permits the development of universal system software, and simplifies training of maintenance personnel. Adherence to a doctrine of machine language compatibility, however, does have the adverse effect of inhibiting the introduction of new techniques and technologies during the subsequent design of replacement machines or additional members. That is, second generation machines may be less efficient because of the necessity to accommodate a previously established word length and instruction repertoire (with all of the implied logic structure) while at the same time introducing newer, more desirable features which cannot take advantage of that hardware required for compatibility.

4.3.2.1.4 Microprogrammed Sequences. The stored logic or microprogrammed technique of computer design offers a third approach to achieving program compatibility in a family of computers. One of the principal characteristics of microprogrammed computers is the basic nature of the micro-operations and, in some cases, the freedom to assign processing functions to a number of general registers. These attributes facilitate "interpretive mode" operation, permitting microprogrammed computers to simulate (or emulate) a given instruction repertoire with considerably more efficiency than would be the case with a "conventional" computer. The benefits of program compatibility at an instruction set level can thus be achieved by equipping a microprogrammed computer (or a family of microprogrammed computers) with appropriate sequences of micro-orders to perform equivalent operations for each instruction in the set.

4.3.2.1.5 Input/Output Compatibility. Like programming compatibility, the ability to control and communicate with a common group of input/output devices is an important characteristic of a computer family. It is not essential that the individual computers have identical I/O instructions but the circuit properties and logic of the interface must agree if all members are to use the common devices.

I/O compatibility is a necessary condition to the use of several different members in a multi-computer configuration with variable or shared peripheral responsibilities and with inter-computer communication requirements. A considerable emphasis on I/O compatibility stems from the relatively high cost of designing and installing special purpose adapters and equipment interface. Also, it is generally true that system growth involves the substitution of a faster, more capable processor along with the addition of more I/O interconnections. That is, there is usually the requirement for the new processor to continue to function with the existing I/O channels. One can envision an evolutionary type of growth, however, in which a next generation processor is equipped with a number of improved, higher speed I/O channels in addition to the normal complement of "standard" channels.

In modern computer systems with memory buffered I/O features, I/O compatibility among family members also implies similarities in methods of memory communication. In particular, if two or more processors are to share common external memory modules with assigned I/O channels, standardized memory communication is required.

An additional memory related aspect of I/O compatibility is the recognition that in order to make efficient use of memory for data storage, memory word length should be equal to the I/O channel width or a simple multiple thereof. This integer relationship eliminates program steps and avoids compromising I/O transfer rates because of the need for programmed formatting.

4.3.2.2 The Family Concept vs. The Unit Computer Concept. The concept of the "unit computer" is implemented in the NTDS. This concept is based on the selection of a basic computer module designed as a fit for the smaller applications. For larger applications, multiples of the unit computer are used. The essential idea here is the use of identical modules. This is the basic philosophy for the Autonetics computers described in Appendix A.

The family concepts described in the foregoing section anticipate a gradation of capability from one computer to the next member in the family. This concept is shown in the design of the IBM System/360 and in the Litton family. The ratio for the Litton machines is approximately 2:1. Complete coverage of the product line is strongly motivated by sales considerations; and therefore, it is probable that a finer gradation of family members is designed than would be dictated by economical considerations only. It is therefore likely that if the family of computer concept is adopted (as opposed to the unit computer concept), the gradation could be more gross. For Marine uses, this suggests a maximum of two computer capabilities with a ratio of from 3:1 to 6:1 between the family members.

Although both concepts are candidate, the distinction between the two is relatively minor.

4.3.2.3 Interservice Compatibility. There is an extensive interchange of data among the services. The need for input of supporting data between services needed for carrying out operational tasks and for updating common files provides a strong motivation for commonality of equipments. The family of computers concept is strongly reinforced by the economics of equipment commonality and it may be anticipated that a common services computer family will be well established in the MTACC era. This is highly conjectural, since it depends on political as well as technological and economic considerations. However, it should be recognized that compatibility with equipment of the other services may be as important as compatibility with equipments especially designed for the Marines. Moreover, compatibility with widely available commercial equipments may be as important as compatibility with military equipment.

4.3.2.4 Relevance to MTACC. To simplify programmer training, minimize program development costs, and facilitate system growth, some form of program compatibility should be achieved. While it is fairly certain that a common symbolic language should be adopted, the necessity of machine instruction level compatibility is not so obvious. The important compatibility consideration is that the members of the family are able to execute the same set of instructions and produce identical results.

The practical requirement for evolutionary change in any installation of equipment places relatively more importance on I/O compatibility. The expectation of multicomputer configurations further underlines this emphasis. Also, the anticipated pattern of system growth, in which a new more powerful processor must work with previously installed I/O equipments, requires careful attention to this family characteristic.

## 4.3.3 Organizational Conclusions

Organizational considerations involve interaction of processors with each other (multiprocessing), with memories, and with peripheral equipment. A number of conclusions in each of these areas follow:

## Multiprocessing

1) The capability for parallel processing should be provided in the form of multiprocessors. The processors should have the capability to attach additional memories and to signal other processors and exchange status information.

2) Shared storage among processors is a desirable feature and may also be required for standby and backup purposes in order that the standby programs and the essential data base may be quickly obtained.

3) Processors should have private (local) memories but should also have the capability to form memory expansion by the incremental addition of memory modules. There should also be provisions that memory modules may be shared to obviate the need for bulk data transmission between processors.

4) There should be provisions for inter-processor communication for transfer of status information and to provide inter-processor health checking.

## Memories

1) A memory hierarchy of at least three levels is required.

   a) Fast scratchpad memory

   b) Fast memory

   c) Bulk memory

   Provision should be made to allow shared memories. However, each processor is expected to have at least one module of local (private) memory.

2) In addition, specialized memory devices may be used in the memory hierarchy.

   a) Content - addressable associative memories. Small modules in the general size of 1024-4096 may be used in conjunction with mass memory devices for such application areas as information retrieval and radar correlation.

   b) Read only memories of two types:

      (1) Large ROM for storage of programs and bulk data which is not subject to change.

(2) Small ROM for specialized uses such as:

- Storage of tables
- Function generators
- Code conversion
- Interrupt processing registers
- Storage of executive routines

The performance characteristic may be fast read, slow write, rather than read only. For such designs it may be useful, for security purposes, to establish a class of privileged instructions with the write capability.

3) Scratchpads. Used to bridge the gap between logic speed and main memory speed and as pseudo-working registers, and for storage of important and frequently used parameters.

Consideration should be given to the possibility of sharing the special memory devices among processors — both from the standpoint of economics (improved hardware use) and from the standpoint of improved capability (e.g., decreased need for interprocessor communication, common functions made available to each processor, etc.).

4) Peripheral and I/O. The computers should be highly interrupt oriented to provide a flexible and self-adaptive aspect to the system. The topology of communication should feature alternate paths to peripheral equipment.

Non-ambiguous data formats and standard codes for use at I/O interface. Standard interfaces should have open-ended design, however.

## 4.4 SOFTWARE TRENDS

### 4.4.1 General

In the areas of software and programming, a number of important trends can be identified which are likely to endure through the next decade and beyond. These developments will provide meaningful and important implications for MTACC. The discussion of software trends is divided into three major subsections. These are introduced below and discussed in greater detail in the following subsections:

1) Computer Languages. There has been a growing trend toward even greater usage of higher order compiler languages as opposed to the machine oriented assembly language.

   Efforts are underway to combine the most powerful features of current compiler languages into what might be termed a universal language.

   A design concept of several advanced compilers is an open-ended design which permits inclusion of new features in the language as the need is seen.

   Attempts to achieve economy of effort with respect to compiler writing are underway with use of meta-linguistic techniques.

2) Systems Support. A dichotomy between the operations programs written by the system user and the systems programs (programming tools), such as the executives, monitors, input/output, and utility packages is increasingly evident.

   The purpose of such system support is to relieve the user of a number of detail computer considerations (e.g., I/O programming), and to achieve efficient scheduling and equipments allocation.

3) Time-Sharing and Man-Machine Relationships. There is a strong trend toward the design of multiprogramming and time-sharing systems. Provision for multiprogramming will be a requirement if these concepts are used in MTACC design. Time sharing is likely to be used at some echelons which use remote input devices. Many of the techniques which are being developed to a greater degree of sophistication are directly applicable to MTACC design.

Implicit programming is a new concept. It encompasses a vast area of investigation and experimentation that will eventually allow the commander and his staff to converse directly with the computer without an intermediary (such as a programmer) to obtain response from the system that has not been preprogrammed. Progress in this area is significant, but the applicability of these developments appear to be extremely limited for MTACC because of the complexity involved in the implementation.

## 4.4.2 Computer Languages

In the context of computer discussion, a language is defined by a set of symbols and a prescribed set of rules governing the manner and sequence in which the symbols may be combined. These languages are members of a class of formal systems of expression similar to the equations of mathematics. Although the power of programming languages is limited, their utility must not be underestimated. By providing a powerful notation, they allow a programmer to concentrate on a method of problem solution, rather than on the problems associated with organization of machine instructions.

The topic of computer languages has an important if somewhat indirect significance to MTACC. Current efforts to develop programming tools to provide more direct rapport between the programmer and the computer give promise of decisive increase in programming effectiveness. In some instances the necessity of assistance from the programmer professional will be obviated and the programming function may be assumed by the application analyst.

An important distinction to be made is that between programs which are intended to be enduring (used on a recurring basis), and those which simply pose a problem for computer solution. For a program which may be used thousands of times, a good deal of program optimization may be justified. In the case of the program written to solve a a particular problem which is not used subsequently, a sophisticated program is not needed. Ideally, the services of a programmer to translate such a problem to the computer should not be required.

A further division among the programs which are expected to "endure", may be made between programs which are used in the field and those which may be needed for system support, such as the programming tools ordinarily available at a programming center.

Programs in the field have the characteristics of being highly refined and optimized according to the most crucial need (e. g. , speed, storage). It is usually found inadvisable to modify such programs in the field. The programs used in the programming centers to support the production programming efforts have no particular tactical significance but have important economic ramifications, since the efficiency of such programs will exert a continuing effect on system costs as long as they are used.

Although the goal of a universal language is still nurtured by many, there will continue to be a multiplicity of higher order languages. PL-1 (formerly NPL and MPPL) being developed by IBM for the System/360 may become a de facto standard because of the vast impetus that will be provided in the installation of System/360 equipment. PL-1 comes close to serving as a universal language, since it incorporates FORTRAN, ALGOL, and COBOL features, as well as being intended for use by the systems programmer. It has features for handling executive and interrupt functions and is appropriate for real-time programming.

Other languages of particular interest to the Marine Corps and MTACC designers are JOVIAL, NELIAC, and CS-1. These languages and others are briefly described in Appendix D.

4.4.2.1 Compiler Languages. The most efficient translation from problem statement to computer program in terns of programmer effort is by means of higher order languages. It is important to maintain the distinction between the language and the language processor. The language processor is the routine which translates it into basic internal machine language, orobject language. This distinction must be kept in mind although the language/language processor pair must frequently be discussed at the same time. Before proceeding to a discussion of existing Language/language processor pairs, language processors in general will be briefly considered. An important type of language processor is the "compiler."

The compiler accepts an entire program and translates it completely into internal machine language. Sometimes intermediate languages are used with two or more translation steps. The translation is preserved for execution at any subsequent time.

The programming costs in computer systems are consuming an ever increasing portion of overall implementation costs. Therefore, ease of programming is often regarded as more important than program efficiency.

The use of compiler languages is generally less efficient in terms of program efficiency than programs produced with machine languages directly. Machine language programs generally occupy less memory and require less computer time for their execution. As a rule, the more general purpose the programming language, the more inefficient it becomes both from the standpoint of the computer time (required by the language processor) and the efficiency of the machine language produced. On the other hand, programming languages are extremely valuable because of the ease with which programs can be constructed, checked out, and maintained.

4.4.2.1.1 Comparison of Compiler Languages. Comparative evaluations between compiler languages are difficult since there are not completely valid bases for comparisons. The basic difficulty in investigating language efficiency is that recognized languages exist in many forms. Even where an agreed definition of a compiler language exists, the compiler programs which implement the program translation are not standard.

Other variables which tend to confuse evaluation efforts are the variations in the operating systems and the types of users. An analogy could be made to natural languages. It is sometimes suggested that French is the most expressive language for novels, and that German is best for mathematical expression. However, far more important variables are at work in determining the quality of expression.

In the case of compiler languages, one of the important variables in addition to the choice of language and the writer is selection of the operating system. In some cases, the supposed advantages (or disadvantages) of a language actually reflect the power of the operating system under which the compiler and compiled program operates. The operating system consists of the set of support programs designed to control the computer as it proceeds sequentially through a string of jobs. Examples of operating systems are the FORTRAN Monitor and NELOS (used in conjunction with NELIAC).

Summary descriptions of FORTRAN, COBOL, ALGOL, JOVIAL, CS-1, NELIAC and PL/1 languages are contained in Appendix C.

4.4.2.4 Metalinguistic Techniques. The development of meta-assemblers and metacompilers give promise for decisive improvement in programming effort effectiveness. These tools provide leverage for the production of other tools at a cost an order of magnitude less.

A meta-assembly program provides the capability to produce code for any conventional (Von Neuman) digital computer. This is done by specifying the characteristics of the computer in terms of the instruction repertoire and instruction formats.

Although the primary purpose of the meta-assembler is for assembling data, it has other powerful uses as well. With the meta-assembly program, the user can define his own pseudo-operations (macros) and can thus tailor his language to his needs.

Another important feature is its ability to produce code for computers other than the computer on which the meta-assembler is running. The systems programmer describes to the meta-assembler both the source language and the instruction format of the target computer.

In case of changeover from one computer system to another, it is possible by these means for the user to prepare his programs even before his new computer is delivered, using his currently available system.

This type of programming tool has important implications for the translating of program libraries from one assembly language to another. In any situation where conservation of program inventories is an important consideration, this tool would find important application.

4.4.2.2 Monitors. Monitors were first conceived for use with off-line computer systems; that is, with computer systems for which inputs and outputs were entirely under the control of the central processor. In most computer installations it is not unusual for the computer time required for the execution of a single task or run to average less than ten minutes. The majority of these runs are used for program checkout and require less than a minute. The time between tasks, if each task is set up after the last is removed from the equipment, will average at least five minutes. Monitors provide the means by which a group or batch of tasks can be executed following a single setup. For this reason, the computers in these installations are called batch processors and the operation under a monitor system is called batch processing.

Even with a monitor system, the central processór of a large system will be idle much of the time awaiting the completion of input and output operations. A recent innovation involves a smaller satellite computer connected to the larger computer. The monitor resides in the smaller computer which assembles inputs for and distributes outputs from the larger computer. The work is processed continuoutly — so that it is no longer a patch processing system but has evolved into a "continuous-flow" system.

4.4.2.3 Symbolic Assembly Languages. Symbolic assembly languages are currently the most universally used computer languages. In general, such a language provides a means by which the instructions for a particular machine may be written symbolically. That is, locations in memory may be assigned alpha-numeric names, and the machine instructions may be referred to by mnemonic symbols. Every useful machine instruction is assigned its mnemonic symbol. Numbers may be written in decimal notation. The advantage of writing programs in an assembly language is that the programmer can maintain almost complete control over every detail of the operation. The disadvantages of using an assembly language is that every detail of the operation must be specified. As a result, the process of programming in an assembly language is time consuming and the programs produced are difficult ot read or interpret because the statements can seldom be grouped into sequences which are very meaningful to human beings.

A modern technique is the "Meta-Assembler" approach. In this case, the characteristics of the object machine (the machine on which the object language is to run) is considered as input data to the processor. Thus, the processor can produce from the same source language, object code to run on any one of several machines.

## 4.4.3 System Support and Executive Control

The programming system development may be thought of as containing two elements: system support programs — executives, I/O packages, monitors, utility programs, etc., and the applications programs which are essentially the working programs. These are sometimes referred to as the object programs, but since this term refers also to the object/source dichotomy used in the discussion of programming languages, it is perhaps preferable to refer to such programs as operations programs.

4.4.3.1 Executive Control. The program which is the master controller of all other programs is called the executive. For MTACC in the general case, computer systems will be required to operate under continuous program control. This dictates that an executive function is required regardless of whether it is a one-computer or a multiprocessor system. The nature of the executive concept may vary, however, at different echelons and for the different orientation of the various processing tasks. For example, an executive program which is oriently almost entirely to message handling and the processing of message requests may appear quite distinct from one which is intended for information retrieval and data management. However, to provide framework within which the operations programs can operate in a uniform manner, the executive program should be constructed in such a way as to accommodate the applications area. It should be sufficiently generalized in design to provide for addition of functions with no or minimal reprogramming.

Executive programs relieve the programmer of the intricate problems associated with the interface between the computer and the devices to which it is connected. They provide overall system control. They minimize the changes which must be made in the computer programs where the system is used in a different way or hardware components are added.

Executive programs relieve the programmer of the intricate problems associated with the interface between the computer and the devices to which it is connected. They provide overall system control. They minimize the changes which must be made in the computer programs where the system is used in a different way or hardware components are added.

Other functions commonly performed by executive programs are listed as follows:

1) Scheduling — The scheduling and/or sequencing of the tasks the system is to perform on the basis of manual inputs, external interrupts, predetermined internal sequences, or priorities.

2) Facilities Allocation — The allocation of the system hardware to the tasks which are to be performed.

3) Real-Time Control — The coordination of system activities and real-time data handling requirements. Real-time data has the distinguishing characteristic that the computer inputs and outputs are determined by other system components. Real-time control involves the acceptance, analysis, control, coordination, and response to real-time data.

4) Data Buffering — The control, temporary storage, queueing, and movement of data through the system.

5) Diagnostics — The automatic detection of errors and malfunctions and the automatic execution of malfunction procedures.

6) Restart — The automatic storage and updating of the files which must be saved in order to restart the system after the malfunction of the system together with the programs required to restart the computer following its repair.

7) System Readiness — The automatic, periodic initiation of those procedures which are required to determine the operability of various system components.

8) System Performance — The automatic gathering, recording, and analysis of parameters needed to evaluate system performance.

The executive for a single computer at a low echelon may provide simply a sequencing function calling for program modules in appropriate order. In a multiprocessing system, the executive must also allocate peripheral equipments and memory moduels, and resolve scheduling problems for possible conflict. It must assure that two processors do not modify data at the same time in a shared memory, for esample. A design goal for executive programs in multicomputer systems is to permit the user (the programmer in the programming center, or the operational user in a tactical situation to be unaware and to make no allowance for the fact that the system has more than one processor.

4.4.3.2 Priority Considerations. The dynamic entrance of priority tasks must be deferred or suspended. A great deal of complex analysis may be required to determine which tasks should be deferred or suspended in order to cause the minimum disruption of established schedules. Saturation of the computing system with job requests also creates a requirement for analysis and judgment in scheduling. For example, a decision is required as to whether all tasks should be a little bit late or one task should be very late in order that the other tasks can be on time. However, relatively simple rules can be invoked which state how the executive programs should handle the scheduling and equipment allocation tasks, allowing for empirical optimization by the programmer through his priority and task definition, and at the same time provide for future improvements to be made which reflect more mathematical or sophisticated treatments.

There are a number of criteria for determining priority. Input requests assume a high priority because of priority possible loss of data. High priority is also accorded failure or malfunction condition signals. The computer, through its supervisory control, automatically monitors the status of current tasks and may dynamically change the priorities as necessary.

4.4.3.3 Input/Output Control System. The computer programs required to read, write and coordinate the operation of input/output equipment are some of the most difficult to design, program, and checkout. Such package concerned with the interface between the computer and any other devices completely under its control is referred to as an input/output control system.

Input/output control systems represent a careful balance between the constraints imposed by the hardware, and the constraints imposed by the programs which the computer executes. They are both machine and application dependent, but are designed to permit the programmer to exercise a maximum of control with a minimum of effort.

A general input/output control system will read data, write data, check for erroneous data, check for malfunctions of the peripheral devices, check for computer malfunctions of the peripheral devices, check for computer malfunction, maintain a record of the status of peripheral equipment and execute standard malfunction and error procedures. Erroneous data checks will include checks for error codes (parity, redundancy, error correction, etc.) and format (too much data, too little data, numeric, alphabetic, binary, etc.). Peripheral equipment status classifications are those for connected, disconnected, operating, malfunctioning, etc. Standard malfunction and error procedures may be automatic (reread or rewrite following the detection of erroneous data) or involve the operator (restart after peripheral equipment malfunction).

In existing systems, the execution of input/output control system programs represents an excessive and unavoidable overhead. These programs require both computer memory for storage and computer time for execution. By applying the modularity and economic special purpose components afforded by advances in hardware technology together with an adequate overall system design, it will be possible to so standardize the input and output procedures that such programs can almost be eliminated. The amount of programming and computer time under actual operation devoted to input/output control provides a criteria for effective system design.

4.4.3.4 Program Checkout and System Test Tools. Good debugging and program testing tools, when they are properly used, reduce immensely the computer and programmer time required to checkout programs. The availability of adequate tools for system test is imperative for the implementation of large systems.

The functions of computer program checkout tools are to provide programmers with the ability to examine the results of the execution of his program in the minutest detail. The concensus is that the basic program checkout tool provides the ability to obtain, at specified points during the programs execution, the status of specified portions of main memory.

It is advantageous if the presentation of the memory status takes a form which is as close to the source language of the program as possible. Numbers should be presented in decimal and instructions symbolically. To achieve this capability, there must be a close relationship between the language processor and the checkout tools. One method of obtaining this close relationship is to embed the higher language processor within the checkout package.

Program checkout tools are useful only for the checkout of individual programs. In order to test time-shared or multiprogrammed systems, special system test tools are required. Even though two or more programs may operate properly when executed independently, the timing relationships and the facility allocations may be such that neither operates in parallel.

A good set of system test tools will have three components. The first will be capable of generating test data with specified characteristics. The second will be capable of driving the system with these inputs and recording the system outputs. The third component will be capable of analyzing the outputs for anomalies. It must be possible for all three components to be executed concurrently.

For the MTACC time period, on-line checkout will be of primary importance. This method was prominent in early computing efforts, but was soon proved uneconomical, since it required the computer to be paced at the slow rate of human decision and reaction. This is a strong revival of this method brought about by time-sharing techniques which permits the programmer to be on-line but using only a fraction of the computer capability. The economics of this usage are not clearly proven at this time, but it can be anticipated that for the MTACC era, checkout of programs will be primarily by this method.

## 4.4.4 Time Sharing and Man-Machine Relationships

An area of computer study which has had increasing emphasis in recent years is the simultaneous use by several users of a computer for different tasks. The development of sophisticated time-sharing techniques has strong economic implication to program production methods.

A computer is said to be "time-shared" when a special programming technique is used to share the computer's central processor among a number of tasks. This technique called "multiprogramming" has the following characteristic. A user's program is being executed by the central processor. At some point in time, before completion of the user's program, the central processor stops executing the program and starts executing another user's program. At some point during the execution of the second user's program, the central processor again stops and either resumes executing the first program or goes to a third user's program. This process continues until the user's tasks are gradually all completed.

At a gross level of detail, time sharing is not new. For many years computer programs have been designed to use a processor for one task while Input/Output equipment is engaged in other tasks. Also, real-time systems have been designed to use a program which is capable of commutating rapidly through many sub-programs. The newer aspects of time sharing are the accomplishment of apparent simultaneity of computer programs by much tighter interleaving of individual programs, and with appropriate protection features to prevent one program from interfering with another.

Time sharing of general purpose computers relies on both hardware and software for its implementation. Early systems had a deficiency of hardware features, placing an added burden on computer programming. This has contributed to a somewhat distorted picture of the complexity of time sharing. On the other hand, it would be unrealistic to assume that added hardware features will eliminate the need for time sharing software.

The general term "man machine" implies any on-line human operated system. In the previous statement the user is assumed to be the programmer. However, the same technique can apply to the military user at the remote console to facilitate decision making. Non-programmers may also program data by providing a simple means of problem expression not requiring extensive training.

Man-machine relationships will be vastly improved in the next decade. This improvement must come almost entirely in the area of the machine since the amount of information which may be recognized, absorbed, isolated, or otherwise processed by man is not likely to increase measurable. However, man is endowed with a combination of processing abilities which will not be successfully simulated except to a limited extent. The judgment, intuition and guidance supplied by the commander will still be required to direct the processes of computation and to effectively exploit the available but cheaper logical power.

The user of conventional data processing centers cannot interact with the program while running. The response time (termed turnaround time) is in hours and days.

Time sharing permits the user and machine to interact with consequent savings in time and programming and costs. Time shared access to computers permits the user to directly request system programs. The user may also incorporate program modifications as needed. At the same time, other users are accommodated.

Apart from man-machine considerations, there are similar situations which stand to benefit from time sharing. Wherever appreciable excess computing capacity exists, other programs can, in principle, use the remaining capacity. It is this improvement in efficiency which is the principal benefit to be derived from time sharing. This benefit is similar to the benefit obtained in communications by the use of compression techniques to better use the available bandwidth.

4.4.4.1 Multiprogramming. The disparity between Input/Output and processing speeds is significant and will become worse as computer speeds continue to increase rapidly and Input/Output speeds increase only modestly. In many systems this situation poses a difficult problem. It is not uncommon for half the processing capability to be wasted due to the necessity for waiting for Input/Output service. To attack this important problem, the multiprogramming technique has been used.

Multiprogramming is defined as the time sharing of a processor to operate concurrently several independent program tasks. Multiprogramming minimizes delays caused by processors awaiting Input/Output service. In addition to input/output/processing load smoothing, multiprogramming exhibits advantages of greater throughput, reduction of turnaround time, improved machine utilization, and less set-up time (as amortized over the greater number of users served).

There are two anticipated environments of multiprogramming usage. The first is in the programming center where program production is the primary activity. The second is in the field situation for which the computer programs have been prepared and checked out and are operating in response to the tactical situation. The users are different, and therefore the nature of the multiprogramming is different.

For the programming center, the user is the programmer, and the operations programs may or may not be checked out. The objective of multiprogramming is to achieve hardware efficiency and improved turnaround time. The memory protect function which maintains program integrity is essential in this environment, because programs are not always completely error free. The trend in such centers is to interpose an operating system between the programmer and the computer. This consists of a large set of programming aids operated under control of a set of supervisory programs.

In the tactical situation the nature of multiprogramming is considerably different. The user in this case is not a programmer. Programs are shared on the computer not for efficiency and reduced turnaround time, but for purposes of tactical priority operation. The programs are called as the tactical situation demands. Operable and operating programs obtain specified responses to messages and other real-time input in an on-line operation. If many programs operate on the same computer in this type of situation, does this constitute multiprogramming. This is best resolved by considering whether more than one user considers his requirements are serviced.

The executive program in this environment varies. A simple sequencing of the programs in a fixed order which is predetermined and for which Input/Output servicing is completely defined is one. Other sophisticated supervisory programs dynamically alter operations programs. These are ordered in accordance with current exigencies. However, this is not necessarily done in accordance with a predetermined sequence.

The executive overhead will vary accordingly from 5% to 25% of the computer program execution time.

There are a number of features, both programming and hardware, to be considered in connection with multiprogramming. The five requirements provided for in hardware and/or in the programming system are as follows:

1) Memory protection
2) Program and data relocatability
3) Supervisory program for Input/Output control and interrupt processing
4) Interrupt system
5) Symbolic addressing of Input/Output

An elapsed time clock interrupt could be added to distribute computer time among time-sharing programs (hardware), and to provide a clean subroutine linkage method for standard use (programming).

A goal of multiprogramming systems is to allow each user to consider that his program is operating continuously; and to relieve him of consideration of Input/Output details. Therefore, contact with Input/Output handled by the supervisor program and user references should be symbolic.

4.4.4.2 Implicit Programming. The realizable hopes for implicit programming are often intermingled unintentionally with expectations which are quite unrealistic. Several properties and capabilities associated with the term follow. The concept provides the following capabilities:

1) Permit a non-programmer technical person to perform the programming function by providing him the necessary tools for communication and interpretation.
2) Revise data formats and display characteristics on line.

3) Allow man-machine communication in near-natural language.

4) Permit the program sequence to be dynamically altered according to real time inputs.

5) Receive answers in real time to questions which have been previously framed.

6) Provide freedom from the usual limitation of a finite number of working routines.

Several of these goals (namely items 2 and 4) have been achieved in various degrees by several time-sharing systems. However, realization of several of these goals is likely to prove elusive. In item 6, the number of system programs is likely to remain fixed, even though indefinitely expandable. One or more programs may be, in effect, a program generator which may in turn produce an object program that is executed (perhaps sufficiently rapidly to be considered as an instantaneous response). The desire to obtain (perhaps vital) answers to questions which have not been posed to (or programmed for) the computer heretofore is realizable only if it is understood that the rules for obtaining the answers must have been specified to the computer in some way. The answers may be deduced from data relationships, by a new progression of routines, by a program generator, or by combinations of these.

The basic limitation is that no computer organization has yet been devised which does more than it is "told", and whether this instruction is fixed in construction, conventionally preprogrammed, or programmed in real-time, does not change the basic fact.

Implicit programming, as seen by the programmer, does not appear as an intrinsically new concept. Rather, it appears as a variation of long-held opinions concerning worthwhile goals regarding programming in a modular fashion such that the current user can select a sensitive and flexible set of routines to satisfy his momentary requirements.

Today, one of the most promising examples of an implicitly programmed system is the display-oriented console system. With systems of this type, the user can request or store both alpha-numeric and graphical displays. He is thus able to receive from and provide to the computer, information concerning relationships. In addition, because of the intimate man-machine interaction, the computer can be used to prompt the user with requests for information and to inform him immediately of inconsistencies and deficiencies in the information he provides.

The limitations for permitting non-programmers to do programming should be recognized. Such goals are realizable and have important economic implications. The goal of such programming is to obtain answers directly without expensive programmer translation; the goal is not, however, to write optimized or enduring programs.

The nature of the language for on-line communication presents an interesting challenge. Ideally, a nearly natural language would be desirable; however, the program structure to support this goal is currently unattainable. In the next decade, the best results will be obtained by keeping the language sufficiently limited in terms of input variables, that meaningful combinations of operations may be specified.

The limitation may be illustrated by considering the inherent difficulty of chess-playing programs. In certain programs, it is estimated that to anticipate one more move ahead (e.g., 3 rather than 2) would require a thousand-fold increase in the current computer capability. Such difficulties are mitigated to some extent by delimiting filter routines that attempt to eliminate large segments of unlikely continuations. These efforts add to the programming structure and the aggregate of such routines will often surpass the complexities of the basic program.

In an analogous way, provision for too large or varied a vocabulary would require a complex structure of special purpose routines to interpret the language interrelationships.

The key to the problem is in the allocation of functions to the man and machine with each performing those functions which constitute its (his) forte. For the man, this is likely to be in the areas of pattern recognition and decision-making. The "chess playing" program that we would like for our military programming chores might consist of a program which when requested would present us with a number of extrapolated choices according to a specified continuation selected by the man. The hackneyed conclusion that computers are best used not as decision makers, but rather as aids to decision making is likely to be valid even in 1975.

# Section 5

# DESIGN CONCEPTS AND APPLICATION CHARACTERISTICS

## 5.1 APPLICATION AREAS AND PROCESSING CHARACTERISTICS

The general functional areas for which computing systems may be profitably applied are:

1) Operations
2) Fire Support
3) Intelligence
4) Logistics
5) Communications
6) Personnel and Administration

The detailed computation characteristics in these functional areas have been studied in detail and are well known. It is appropriate to summarize the basic characteristics of these functional areas, translate these into computer characteristics, and determine the implications on computer design. A study of the span of applications in terms of computer characteristics should give an indication of the type of computer design applicable. In particular, it is important to discover how disparate the problems are and whether a single (general purpose) design or several (special purpose) designs are appropriate.

Assignment of function to computer modules could be made on the basis of the particular type of processing. For example, file management, mathematical processing, or message handling could each benefit from a design specially optimized for the particular functional performance. However, the advantages of this approach are far outweighed by other considerations, primarily in the area of communication limitations. Other compelling considerations which preclude this approach are those of load smoothing, priority considerations, and geographic deployment, which is required to reduce vulnerability and to provide continuing capabilities at alternate locations.

### 5.1.1 Functional Areas

Some of the representative component parts of basic functional requirements for MTACC include the following:

1) Operations
    a) Information Requests
    b) Information Display
    c) Operations Planning Support
    d) Landing Plan Preparation

2) Fire Support
    a) Fire Planning
    b) Target Analysis
    c) Ammunition Accounting
    d) Tactical Fire Direction
    e) Fire Support Coordination

3) Intelligence
    a) Retrieval of Intelligence Reports
    b) Enemy Capability and Location
    c) Terrain and Weather

4) Logistics
    a) Stock Inventory Status Reports
    b) Consumption Reports
    c) Availability of Resources
    d) Requisition Processing
    e) Transportation Schedule Generation

5) Communication

a) Message Transmission
b) Message Format Conversion
c) Validity and Error Checking
d) Automatic Frequency Allocation

6) Personnel and Administrative

a) Personnel Record Maintenance
b) Payroll
c) Unit Status Reports

### 5.1.2 Processing Characteristics

The range of the functional processing indicated in the preceding lists encompass a span of processing activity from highly mathematically-oriented problems to those of data management and information retrieval. Examples of mathematical problems are ballistic trajectory calculations, meteorological predictions, and target interception calculations. The other extreme is typified by message handling and stock control applications. The approximate position of the MTACC functional areas in the range is illustrated in Figure 5-1.

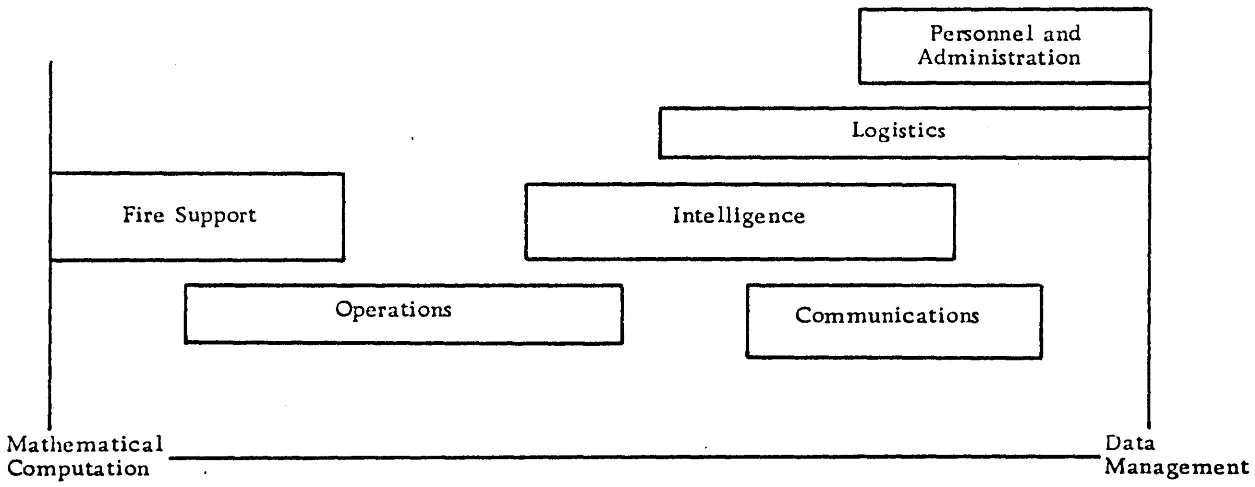


Figure 5-1. Processing Requirement Characteristics Span

The functional areas could also be plotted against other characteristics, such as data volume (amount of input/output), and real-time response requirements. For fire control applications the volume of data is relatively low and the computation requirements high. Logistics and personnel applications have large data volume characteristics but less complex computations.

Other characteristics from left to right in Figure 5-1 are:

1) The computation problems range from very fast response characteristics (real-time) to periodic processing (non-real-time) characteristics.
2) The data storage requirements vary from modest to voluminous.
3) The processing tends to be appropriate for higher command echelons.

## 5.1.3 Information Retrieval

Information retrieval is characterized as a request/response process. The requirement of this type problem is to obtain information in a timely fashion based on a partial description of the desired data. Another characteristic associated with this problem is the dynamic nature of the data. Data ordering, file updating or sorting may be required frequently. A requirement for tactical command and control system information retrieval is that the response patterns may change dynamically in order that the most relevant information be the most readily accessible. In some cases, the response is required almost immediately in real time. Information retrieval for MTACC is particularly characterized by high turnover rate, and need to access using different keys. This presents the difficulties of reorganizing and reconstructing files. This type of processing is too lengthy to be done in real time and therefore a preferable approach is to construct new indexes to reflect changing requirements. An integrated hardware software design (e.g., list processing and associative memories) is particularly appropriate here. Due to the volatility of a great part of the data, the use of read only memories is not appropriate. However, read mostly memories may be useful for personnel and administration and for certain logistics accounting.

There are several approaches to the problem. One approach involves the arrangement of data into a stratified structure which facilitates retrieval. In particular, list structures have been found effective. Further benefits can be derived by the development of special instruction sets to provide the manipulative functions needed.

Another approach is to develop special capability hardware. Since a characteristic of associative memories is the capability to bypass scanning and sorting in many instances, it is often considered for this application.

Another characteristic of the information retrieval problem is the generally voluminous size of the data files. For future tactical data systems, commodious storage capacity will be needed and retrieval of selected data will be required on a real-time basis. The advantage of the associative memory for this application would be twofold: speed of retrieval and a lessening of the requirement for frequent reordering of the data files. The latter consideration is particularly important for problems where there does not appear to be any one best ordering scheme. Since the development of extremely large associative memories appears remote for economic reasons and probably cannot be justified for the MTACC environment, the use of associative memories to contain only a select amount of key data is indicated for indexing into the lower levels of the file structure.

The approaches to information retrieval by programming methods are highly developed. In this area sophisticated programming tools are said to offer an economical alternative to hardware associative memories. In particular, the use of list structure storage and list processing methods have proved extremely useful. Historically, the same motivations which have given impetus to associative memory development have resulted in the development and refining of list processing techniques.

### 5.1.4 Message Handling

One of the primary applications for computers for MTACC is to relieve insofar as possible, the communications requirement. To a limited extent, capacity may be traded for communication capability. This is done by condensing and combining messages, by the sending of only the most important summary information, and by using a "data request" rather than a "forced data" philosophy. This would cause requested data to be transmitted but would tend to eliminate data that is not needed.

Certain computer characteristics are important for the general application of message handling. The essential processing consideration is the capability to accommodate the peak message processing load with little or no delay to essential services. In the area of instruction repertoire design, a capability for bit manipulation, the transformations of fields (for message format conversion), and a strong set of (Boolean) logical commands is indicated. A highly developed interrupt structure is also important, along with the necessary interrupt processing logic within the executive programs.

An important trend in the message handling problem relates to the message content itself. The message handling function in future systems may emphasize interpretation of the message content itself in order to determine disposition and routing. The semantic significance of the message will thus interrelate with the program logic to effect efficient communication. The development of sophisticated techniques in this area will motivate toward a freeing of the communication lines for the highest priority usages.

## 5.2 FAMILY CONCEPTS AND COMPATIBILITY

The concept of a family of computers will be most important for an MTACC system. In the country today the most important decisions facing computer manufacturers relate to problems of compatibility and interchangeability of hardware and software among elements. For similar reasons these questions also face an MTACC system designer. In fact, many of the important questions become even more critical in view of the special operational demands of a future Marine Command and Control System, especially those of compatibility with other services.

It is quite likely that the computer systems in use in a future MTACC system will be similar to or compatible with the system which will be developed for CCIS-70. The concepts and principles suggested here cover the system concept, considerations of implementation of command functions, the software problem, and the question of compatibility. All of these subjects are treated from the point of view of a family concept of computers. The question of relating this to CCIS-70 is ignored in this discussion since there exists no specific computer concept for that system. The presentation here may well fit the choice of a computer system for CCIS-70 if it is a concept acceptable to the Marine Corps and if the Marine Corps can, in turn, influence a decision on CCIS-70.

Decisions relating to the development of a family of computers will affect all evaluation criteria for computers. The concept will affect cost since the degree to which hardware and software modules can be replicated in a fieldable system has direct bearing on the cost. Cost is, of course, a pervasive criterion since nearly all other technical factors directly relate to cost. Maintainability problems are also involved since the entire question of spare parts and training is heavily dependent on a family concept. Last, but certainly not least, operational responsiveness is heavily influenced by decisions on compatibility and interchangeability.

## 5.2.1 The Single System Approach

In the Army FIELDATA System as it was first conceived some eight years ago, it was to be a family of computers. These computers, the MOBIDIC, the BASICPAC, the INFORMER, etc., were in reality different computer systems. There was a degree of compatibility which was achieved mostly, however, through interfacing and communications equipment. The compatibility was restricted to format and electronic characteristics necessary for convenient operation with the communications network. Each of the problem areas in the Army FIELDATA System to which these computers were to be applied were regarded as separate and independent application areas. The computers, as a result, had different charactoristics. In the light of newer technology and the family approach to computers, this concept should not be acceptable either to the Army or to the Marine Corps, or future fieldable command and control systems.

Computer system technology for Tactical Command and Control use must follow many of the lines of development of commercial computer systems, at least with respect to families and compatibility concepts. The overall motivations are the same in both cases. The computer manufacturer wants hardware and software costs kept to a minimum and wants his systems reliable and easily maintained. So does the field commander. Not only is hardware and software compatibility within the major manufacturers' computer lines increasing at a very rapid rate, but amazing strides of progress are being made on compatibility between competing family lines, that is, families representing different manufacturers. In 1956, the first problem oriented language (FORTRAN) came into the picture, representing what was probably the first major step in compatibility. Since then not only problem oriented computer languages, but internal machine languages are in many cases identical. We have also reached the point where there is a hardware module interchangeability and where various modules can be added or deleted from systems to make up a complement of equipment which is responsive to a customer's needs or a certain application.

Proceeding then from the developing picture in the commercial computer area, the extension of the ideas to Tactical Command and Control is simple and direct. The obvious conclusion is to develop <u>one</u> computer system to be used in all applications of a future MTACC system. The single computer system might well be regarded a family of computers since it will be possible to select certain types and numbers of modules to make up an installation. While it is possible that there may be some exceptions required to the single computer concept, the concept is certainly a laudable objective and the exceptions, if any, are not likely to be serious ones. In the following paragraphs, the single family concept is introduced. Prior to this however, some observations should be made which further motivate the approach:

1) In the 1975-85 time frame a modest priced computer will be very fast. Because it is very fast, it will be performing many different kinds of operations in any single installation — receiving data, formatting data, mathematical manipulations, buffering, and the like. It would be costly and inefficient to try to make the computers slower and therefore less costly, since by doing so the capability-cost ratio will drop markedly.

2) From an internal computer point of view, all applications in tactical operations are similar. At any future installation in a tactical system the following kinds of operations will be required:

   a) Receiving data from communication lines.

   b) Processing and formatting data.

   c) (Optional) Mathematical manipulation of quantities.

   d) Buffering and formatting of data for internal operations.

   e) Accessing of data from files.

   f) Formatting and synchronizing data for output to the communications net.

To repeat, <u>all</u> systems will be doing the same general type of operation. It is only the quantity of the processes which will vary from location to location.

3) The system concepts which embrace modularity and multicomputer designs are very much more widely accepted and understood. The hardware designs for accomplishing multipurpose and flexible systems and the software concepts for controlling the systems have not only been developed but have been implemented in operational systems.

The single computer concept then is described as having the following characteristics:

1) There is one basic computer module. The module is high speed, has the ability for time sharing, and can be used as a free-standing computer or imbedded in a multicomputer system. Its basic order structure is general purpose. The module can accommodate a wide variety of data processing and computation functions.

2) The basic computer module is part of a modular, multicomputer system. Modules can be added to the system to provide increased overall system capability. The system can accommodate a variety of auxiliary storage and peripheral equipment such as buffers, mass memories, printers, and the like.

3) The system is literally open ended, that is, at any stage of development additional capability can be added without major hardware modifications. This system was conceived ab initio to be open ended. Likewise, the software prepared for the system allows additional applications and modules to be added without changing the basic structure.

4) The system can be used on a highly centralized basis where time sharing is employed or it can be used in smaller installations where fewer applications are involved. Furthermore, the computer can be "pulled apart" during tactical operations in the field depending on the changing requirements of the tactical operation.

The objectives of this family are believed to be almost 100 percent achievable in the 1975-85 time frame. It is possible that there would necessarily be certain minor modifications to the description. For example, for some applications it may be desirable to substitute a higher speed memory in the system in the place of a lower speed memory. The higher speed memory is compatible with the arithmetic control and power supply units. Other relatively minor modifications to the concept may be desirable.

What is being described here is one family, in fact,. one system. The basic application program modules will be used in all systems. Upward and downward compatibility of the system is to be achieved at least with respect to the basic computer module. Application programs will be modular and will interface with executive and input/output systems and can therefore be used in a wide variety of configurations and subsystems.

## 5.2.2 Implementation of Command Functions

Having defined the basic system characteristics of a family of computers, the question arises as to how these principles should be applied to the field organization and functions of an MTACC system. Questions immediately arise such as the following: What is the distribution of the computer modules and systems with respect to the various command levels? What is the distribution with respect to the various functions to be implemented in tactical operations? What kind of command and functional groupings should be implemented by groupings of computers?

To advance general principles of a family of computers application to an MTACC system, it is useful to consider the problem abstractly. Figure 5-2 illustrates a matrix comprised of command levels on the vertical and functions to be implemented by computers on the horizontal. For illustrative purposes, five command levels are shown, one through five, and five functional areas are shown, A through E. The command levels represent ones such as MEF, MED, MEB, MEW, and the functional areas represented, such as fire support, intelligence, ground operations, logistics, etc. Each box formed by the matrix represents, therefore, a function corresponding to a certain command level.

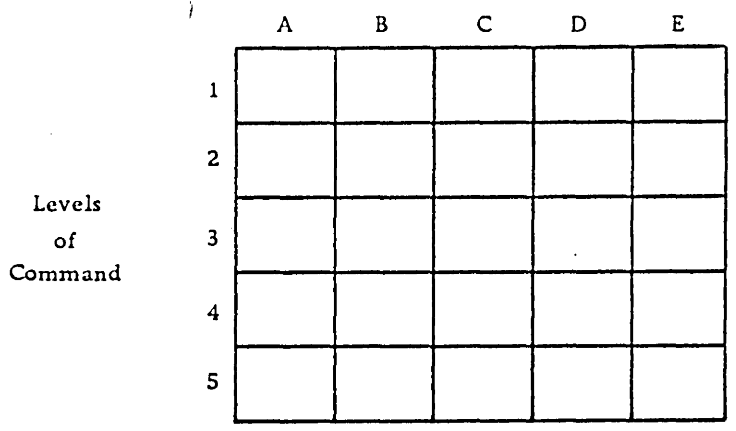


Figure 5-2. Command/Control Functions

Each box of the matrix could conceivably be implemented by a computer. There could be a computer for fire control at the battalion level, or one for intelligence at the Division level, or one for ground operations at the Unit level, etc. Or the boxes may be grouped to a single computer system, that is, both fire control and ground operations, for example, at the battalion level, could be accomplished by a single computer. In addition, certain of the boxes may be implemented by an input/output device which reports upward to a computer or which uses a computer at a higher command echelon.

Leaving aside for the moment I/O implementation of some of the boxes, consider the various ways of aggregating the boxes. (Aggregating in this sense means a grouping of the command-function boxes so that they are implemented by one computer system or one co-located set of equipments.) The boxes can be aggregated horizontally. This corresponds to one computer system for each command level. Obviously, however, location may be a problem. It may be that the fire support computer may not be conveniently co-located with the intelligence activity, for example. It is apparent that if horizontal aggregating occurs on a large scale, then duplicate or standby systems must be considered in order to get the reliability in the case of a direct hit on the computer system.

Aggregation of boxes vertically is meaningless, if not impossible, because of the different locations of command posts, the ground operations functions could not be consolidated into one computer. However, aggregation can take place in a vertical fashion by the use of remote consoles. That is, lower command levels can use computers at the higher levels through I/O equipments which would essentially, therefore, imply time sharing of those equipments.

Certain of the boxes of this abstract representation may not be filled with any equipment. It is doubtful, for example, whether there is a meaningful way in which the intelligence function can be implemented at the multicomputer level. However, here again an input device which is man-transportable which inputs observed enemy locations to higher echelon systems might well be considered. The FIELDATA system implemented the various functions to different levels. The computer implementation of fire control is carried on down to the battalion level in CCIS-70, but computers implement other functions only at higher levels.

Although we have not, in the preceding paragraphs, developed a specific answer to the question of implementing various functions at various command levels, we have, at least, developed a methodology of an approach to stating the problem. Essentially each box of the matrix must be a computer or an I/O device, or the conclusion is made that there is no mechanizable function at that level. The boxes can be grouped horizontally but care must be taken on location and total net reliability questions. Vertical aggregating can be accomplished by I/O devices which input or output data or implement system time sharing. Of course, various combinations of vertical and horizontal aggregating can be accomplished.

### 5.2.3 The Software Problem

In computer based systems, it is commonly asserted that the software costs as much as the hardware. The system responsiveness and the implementation schedule depends strongly on software. The software problem is very closely related to the problems of general compatibility and a concept for a computer family.

The single computer concept can be extended to software, in other words, a single software concept to match this single computer concept. To be more explicit, the following is visualized:

1) Since there will only be one basic computer system, there will only be one machine language used throughout.

2) Since there will be only one multicomputer system concept, there will be only one structure for all executive control software portions. In other words, there will be one system for executive control, I/O program modules will be added to it as necessary. The executive itself will be independent of the number of replicated modules.

3) There will only be one higher order language, or if it appears desirable, there will be one for each basic application of the system. However, each language will be used universally throughout the system. This could be referred to as the question of whether there should be one or more than one language which can only be answered after study. Perhaps the best solution is to have one overall language and have compilers which compile certain subsets of the language. The compilers themselves will, of course, be universal with respect to the set of tactical computers because of the universality of the machine language.

4) In the early development of the system before hardware becomes available, there will be only one set of simulation programs to test out and exercise the various computer program modules. One integrated set will suffice because of the compatibility achieved through the family concept.

There are, however, design problems associated with this overall approach. The upward compatibility of the total software package is an objective which is attainable but not easily so. It will require a very careful design to be able to use program modules of smaller systems in bigger systems where the executive and I/O interfacing program modules will be more elaborate. Also, the design of executive programs which are usable for different types of configurations will not be a simple matter. Again, it will take much design but it is deemed achievable during the time period. The present state-of-the-art is that executive systems can be designed for modular systems which are independent of the number of modules used. The objective here would carry this a step further: the executive program would be usable over a wide variety of systems, from systems which have very few peripherals and bulk memory equipments to those systems where these types of equipments are in abundance. However, much programming efficiency will have accrued by the single system approach and its extension to the software problem even if these problems and these design objectives are not achieved to their fullest.

### 5.2.4 Compatibility

The importance of compatibility in the use of computers in an MTACC system cannot be over-emphasized. All of the factors of lower cost, maintainability, reliability and responsiveness are gained largely through compatibility. In the above paragraphs, much of what has been recorded is directly related to compatibility. However, a subject of such importance as this deserves a more specific and incisive examination.

Compatibility is a much used word in military systems. It is an objective worthy of much effort to achieve but is also is a subject which should be carefully examined. Frequently, systems are constrained unnecessarily in their growth and development by meaningless requirements for compatibility and, frequently, debates rage over questions of compatibility where the compatibility, when achieved, is of little consequence. A minor difference in format, for example, can be solved by the execution of a few extra instructions, a small penalty to pay in view of possibly far more expensive alternatives.

There are three degrees of compatibility which should be considered with respect to computers in an MTACC system. The first of these is compatibility within MTACC, the second is compatibility with respect to the systems with which it must communicate or the systems with which it is similar (such as shipboard ANTACCS systems or land-based CCIS-70 systems). The third area is general compatibility within the Department of Defense. The following remarks are made about each of these types of compatibility:

1) Intra-MTACC Compatibility. Compatibility within the MTACC system itself is highly desirable and can be achieved at relatively little cost since the system will be developed as a system by one service. Modern technology has progressed to a point where reasonable system planning and system design can achieve a very high degree of compatibility within an MTACC System.

2) Compatibility with Respect to an ANTACCS System and the CCIS-70 System. Compatibility relative to ANTACCS will be important from a number of points of view. First of all the system will communicate with the ANTACCS system because the chain of command for the Marine forces will likely be through the Naval command hierarchy. Also, on board ship before command and control has been transferred to the landing forces the command and control system will operate in conjunction with ANTACCS type equipment. The question of compatibility with CCIS-70 arises from the probability that there will be much communication between the Marine system and the Army's system. In many cases, in fact, the Marine command and control system may operate in conjunction with the Army system when the transfer of the battle direction from the Marine Corps to the Army is being achieved. Since the CCIS-70 operation and objectives are very similar to those of an MTACC system, the Marine command and control system may well evolve from the CCIS-70 system and have many equipments and procedures in common with that system. In other words, the compatibility with respect to the CCIS system may be all-pervasive.

3) Compatibility Within the Department of Defense. It will behoove ANTACCS planners, of course, to make sure that the data handling equipments of the future ANTACCS system will be compatible with DOD equipments and

> procedures wherever that compatibility has been defined. However, it is unlikely that any overall DOD compatibility dicta, other than software-type ones such as language compatibility and communication and data format, will be achieved in time to affect any 1975 system.

The advantages of compatibility are generally well understood. However, for the sake of completeness they are discussed in the next few paragraphs, both with respect to hardware and software.

Concerning hardware, the important advantages of compatibility are in spares purchasing and stocking. The more hardware compatibility which is achieved, the greater the advantage. If, for example, the same spares which CCIS-70 uses are also to be used by an MTACC system, the spares could then be achieved frequently through Army channels, if this is made possible through procedures within the Department of Defense and if the geographical location allows it. If hardware compatibility is achieved maintenance procedures will be the same throughout large segments of the MTACC system and within large segments of land combat organizations. This will allow standardization of training with the resultant lower cost and higher quality of procedures and training. Another important aspect of hardware compatibility is that development costs will be less and interfacing equipments less costly and, frequently, unnecessary. It would be highly advantageous, for example, if the peripheral and auxiliary storage devices of the CCIS-70 were directly usable in an MTACC system.

The situation, however, with respect to software is somewhat different. While software compatibility within an MTACC system is highly desirable, it is doubtful whether much is gained in software in having software compatibility between various command and control systems. There is, for example, very little trading of computer programs within the command and control systems and therefore in this respect there is not a large advantage in machine language compatibility between an MTACC system and other command and control systems. Even though many of the functions of CCIS-70, for example, will be very similar, the differing environments of these systems and the differing operational requirements, coupled with the lack of standardization from the functional specification point of view, means that there will be little advantage gained through machine language compatibility. However, it is true that subroutines might be interchangeable as well as certain highly standardized processes such as sorting or merging programs.

The controversy which was underway two to four years ago, with respect to standardization within the Department of Defense, of a problem oriented language such as NELIAC or JOVIAL, was probably premature and probably still is today. In the first place, there is an important question of whether <u>one</u> language should be used for command and control. Command and control is a very comprehensive subject which embraces many procedures and processes ranging from highly mathematical ones to highly procedural or manipulative ones. Perhaps the mathematical processes should be programmed in a higher order language which is efficient for use in such programs, whereas a different kind of language should be used for the procedural or manipulative programs. The standardization on a higher order language does achieve the certain efficiencies in training of programmers. However, a much more important factor is whether the machine languages are similar since, if they are not, the different translators must be developed for each language, and much of the language compatibility benefits are lost.

## Section 6

## IMPLEMENTATION CONCEPTS

In this section, many of the basic questions of implementing the system are raised, especially as they relate to the computers. Many important problems and questions are discussed in implementation and tentative solutions are presented. The questions of implementation tie together the technical, administrative and procurement aspects of the problem. The particular points of view discussed in this section are the relationship of the computers to those of CCIS-70, questions of contractor selection and system specification; and the overall time phase procedure which might be followed in specifying and procuring the computer system.

### 6.1 RELATIONSHIP TO CCIS-70 COMPUTERS

The highest priority questions regarding implementing the computers in an MTACC system concerns the relationship of the computer system to those of CCIS-70. There seems to be little doubt that much of the computer technology and implementation concept to be used in an MTACC system will be the same as those for CCIS-70. There seems to be strong sentiments in this connection in high Department of Defense policy circles. Also, the use of an Army system and its adoption and adaptation to Marine uses is not without precedent; it has been accomplished many times in the past with respect to weapon systems and supporting systems.

Questions which arise, therefore, are those which relate to the implications of the similarity of the equipment with the CCIS-70. Questions are immediately raised as to the differences and similarities of the Army's needs compared with those of the Marine Corps. If the Marine Corps does adopt the Army's command and control system, or at least major parts of it, what kind of modifications will need to be effected? Does the Marine Corps have needs which the Army does not?

To what extent can the overall system philosophies be directly transferred to the Marine Corps system? A full understanding of these questions has a very high priority in any implementation concept; the problem is a very fundamental one to implementing the MTACC system.

A cursory analysis of the question of the difference between the Army needs and those of the Marine Corps suggest some very important differences. First of all, there is the matter of greater transportability requirements of an MTACC system. Army requirements can be less stringent for there will be more time and more capable transportation vehicles to initially bring the system to its position of field use. An MTACC system must be in sufficiently small pieces to be lifted from the decks of ships by helicopters. The implications of these greater transportability needs of the Marine Corps needs more analysis and understanding.

There will, of course, be certain functions which must be implemented in an MTACC system which are unnecessary in the CCIS-70 system. For example, the question of Naval gunfire support is necessary and is a high priority item for an MTACC system. Air support concepts will require different computer uses. Since the Marines regard air support as an integral part of their land forces, they have a different philosophy of operational employment. In an MTACC system, data bases are likely to be reduced in size since the Marine Corps is a very highly mobile and smaller organization. As a result the USMC will have less time and less facility to develop large data bases concerning friendly or enemy forces. Another important difference is again related to the philosophies of personnel use. The Marine Corps works in smaller units than the Army, and needs a greater coordination between smaller field units and fire support units. This may make necessary the use of computers at a much lower organizational level for an MTACC system than is currently being considered in the CCIS-70.

Although many of the answers to the questions remain obscure at this point, there are some general principles regarding this question which can be advanced.

1) At an early date the full point of view of the Department of Defense with respect to an MTACC system and the CCIS-70 should be developed with participation by appropriate Marine Corps personnel.

2) Assuming MTACC-CCIS-70 compatibility to a high degree is dictated, Marine Corps personnel should participate with Army personnel in the development and specification of a computer system.

3) An analysis should proceed immediately to the differences in MTACC and CCIS-70 requirements to understand at an early date the points of departure from CCIS-70 systems and modifications which need to be developed, if any.

4) The same analysis would be obtained for joint development or interfacing systems with the U.S. Air Force and the U.S. Navy.

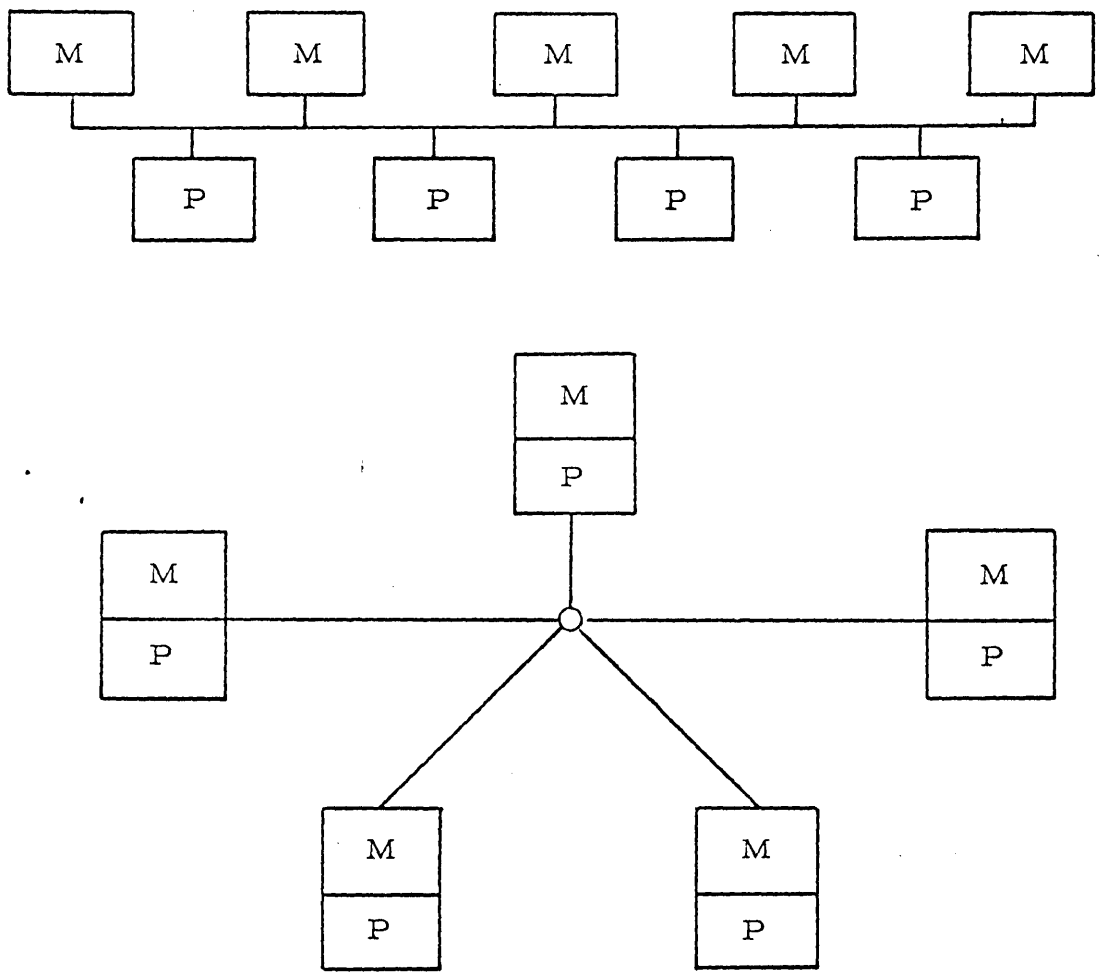
## 6.2 THE CHOICE OF A CONTRACTOR AND A SYSTEM

As stated previously, a number of contractors are qualified to develop computer systems for an MTACC system. There are, however, some important questions which arise as to how the contractor should be selected and how he should proceed in developing the system. There are a number of contractors who are developing proprietary approaches to computer systems. Since these are proprietary approaches, they are aimed at a wide segment of the computer market. They are useful in airborne applications, land based combat operations, reconnaissance, and in fire control. However, this is not necessarily a disadvantage and, in fact, those contractors which have been aiming at a wide military market segment have a broader systems concept and, in general, are in a better position to create a family of computers.

The various computer manufacturers have differing approaches to computer systems. For example, one computer manufacturer may have a system concept where the high speed memory is modular and communicates with various processors. This system concept is shown in Figure 6-1. Another manufacturer may have computer modules where high speed memory is integral with each processor and various processors can communicate with each other. The alternative approach is also shown in Figure 6-1. Although one system concept may be preferred over the other one, the contractor with the least desirable system concept might well be selected because of other factors entering into this decision. In other words, the choice of the contractor will depend on a balance of many factors ranging from the technical ones to more administrative aspects.

Consider some of the various factors which might figure in the choice of a contractor:

1) Contractor experience. Has the contractor had experience developing extensive computerized systems? Has this experience been extended to militarized systems? Has he had experience in developing families of computer systems?



M = Memory
P = Processor

Figure 6-1. Basic System Concepts

2) Production capability. Has the contractor shown ability to produce equipment on time and with the required levels of reliability? Can he take a firm specification and develop a production schedule which is realistic and which will be adhered to?

3) Computer system concept. Does he have a concept for a computer family which embraces the total system and the manner in which it might be used by MTACC? Does he have a concept in mind, for example, that allows modules to be joined together to create computer systems of increasing capabilities? Does he have a comprehensive plan for adding peripheral equipment to the system or for adding higher speed memory or arithmetic and control modules?

4) Systems experience. What systems experience has the contractor had in the specification and evolution of complex computer systems? Has he shown ability in the past to develop a total system concept and carry it through to a usable total product? Has he had the experience in seeing the computer system imbedded in large scale military systems?

5) Basic computer speeds. What kind of circuit and memory speed capability is the contractor capable of implementing reliably? Is the speed of his intended computer system sufficiently high to guarantee a high capability/cost ratio? Is he pressing the technology with respect to speed or is he beind the state-of-the-art technology with respect to speed?

6) Circuits/component design. Is he capable of using the most modern circuits and components in his computer system? Is he using integrated circuits to an advanced degree to allow sufficiently large throw away modules which will generate easily maintained systems. Does he have a good balance between modern circuitry and components which are pushing the state-of-the-art and conservative circuits and components which are sure to yield good results in the time frame desired?

### 6.2.1 Contractor Selection Ground Rules

There are a number of ground rules which can be advanced for choosing a contractor and for developing a responsive system:

1) Develop a nonrestrictive set of specifications which will take advantage of the existing in-house developments of the various contractors, then

2) Let two or more contracts for the development of a computer systems concept, and then

3) Select one contractor and let him continue to develop the system according to the principles specified in Section 6.3 which follows.

## 6.3 IMPLEMENTATION PHASES AND PRINCIPLES

There have been serious shortcomings in the procurement and implementation of procedures used in computerized command and control systems in the past. It is worthwhile to examine these briefly to analyze what has been done and to determine the specific areas in which improvements can be made. This description is covered in the following sections which include a discussion of an implementation procedure which consists of the definition of various phases which relate to computer analysis, definition and procurement.

In the past the following procedures have apparently been followed:

1) Each major functional area of the command and control system has been analyzed in detail.
2) For each function analyzed, detailed computer requirements are determined.
3) Detailed computer specifications are then developed and procurement made on the basis of the detailed specifications.

### 6.3.1 Disadvantages

Some serious disadvantages are as follows:

1) Specifying in detail the computer to be procured, advantage has not been taken of the contractor's in-house developments since he was constrained to respond to the specifications advanced. Extra cost in the procurement was therefore incurred.
2) This has resulted in a basically different computer system for each type of application.
3) The requirements which were considered to be adequate were changed with time as the application area became better understood. The computer therefore in many cases no longer fits the requirements as well as was intended.

## 6.3.2 Alternative Approach

Instead, consider the following approach to computer development:

1) Develop a general purpose computer system on the basis of preliminary requirements and analysis of the technology.
2) Continue application studies to obtain refined estimates of the requirements and the degree to which the computer system meets the requirements.
3) Make continuous modifications and refinements to the computer system as necessary.

In other words, the computer system under this plan would be developed in parallel with the systems analysis and continuing application studies. Even at the outset, before detailed application studies are made, the requirements are known sufficiently well to begin a development of hardware. The hardware then evolves as the system knowledge evolves.

Consider the implementation process previously described in more detail. Figure 6-2 shows the major implementation steps required in proceeding from the preliminary requirements and technology analysis up to the specification of the amounts and types of equipment. The various steps are described in the following paragraphs:

1) Analyze requirements and technology. This is the step being accomplished at the present time with the MTACC effort. The overall desired approach to the family of computers and the desired attributes for field use such as transportability, modularity and system expandability, would be spelled out.
2) Define computer systems concept. This was the step described above as consisting of (perhaps) two contractors working in parallel on two competing design concepts. During this phase the system is described from the standpoint of the modules to be developed, the information flow among the modules, communication within the computer system, the manner in which expandability will be accomplished and the basic approach to circuits and components.

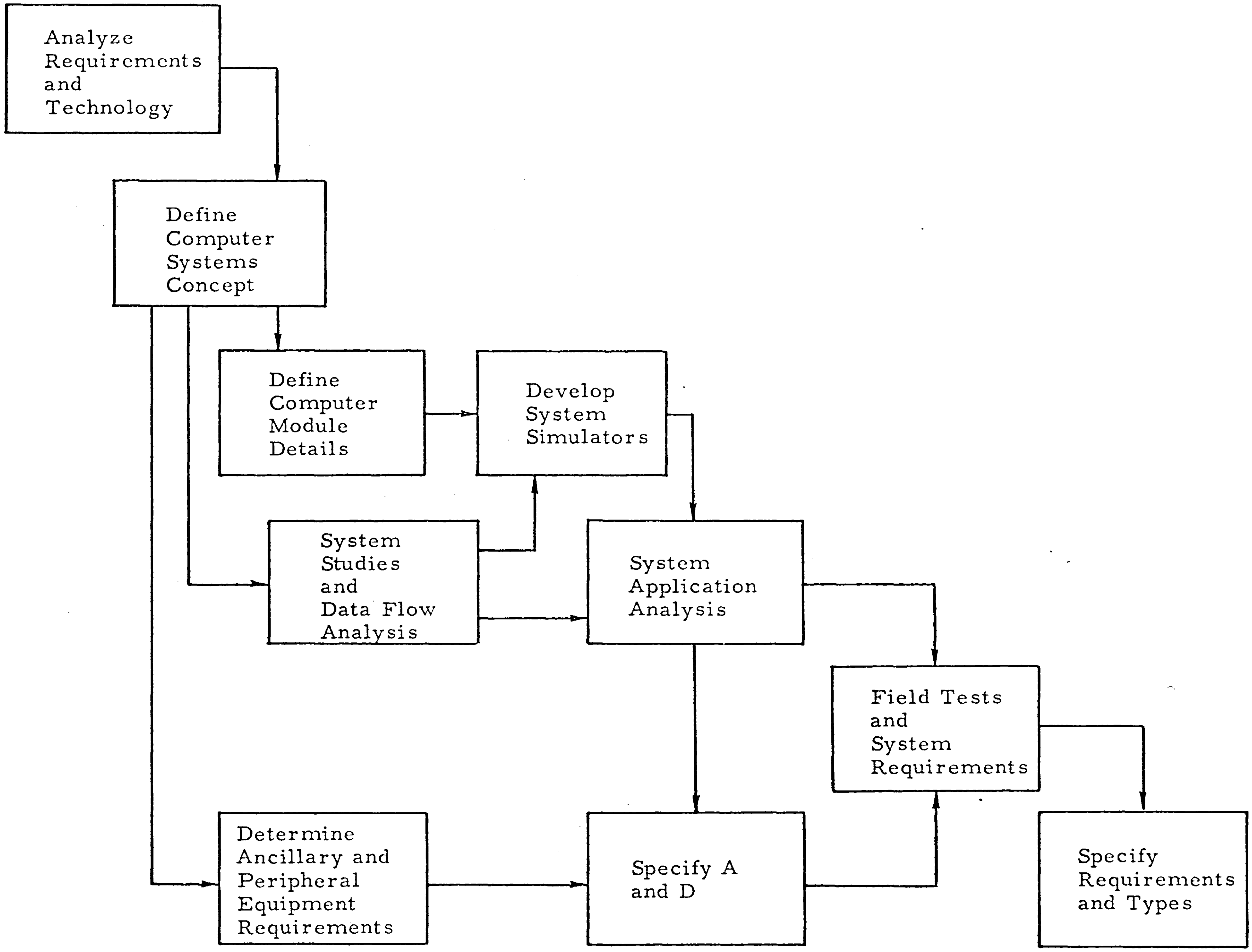

Analyze Requirements and Technology
Define Computer Systems Concept
Define Computer Module Details
Develop System Simulators
System Studies and Data Flow Analysis
System Application Analysis
Field Tests and System Requirements
Determine Ancillary and Peripheral Equipment Requirements
Specify A and D
Specify Requirements and Types

Figure 6-2. Implementation Steps

3) Define computer module details. During this phase the computer module is spelled out in considerable detail. The module design must fit within the system concept described in the previous step. This phase of the work can be accomplished in parallel with the previous step on systems concept or in fact, can precede the definition of the computer system concept. All details of the module are described: its instruction logic, all internal registers, input/output control techniques, interrupt logic and components and circuits.

4) Determine ancillary and peripheral equipment requirements. Based on the definition of the computer system concept and the requirements and technology which have been determined, the requirements for ancillary and peripheral equipments are spelled out. This will include all items such as magnetic tapes, printers, mass magnetic storage, communications interconnecting equipment and the like.

5) System studies and data flow analysis. Based on the definition of the system and the computer module, system studies are accomplished. Sample programming will probably be included in this step to insure that the computer system will work as an efficient integrated whole. The various circumstances will be developed under which the computer system will be expected to operate such as various configurations, various degrees of system degradation, various types of use such as time sharing, and the like. At this point also extensive specifications are developed for executive control programs and other standard software required.

6) Develop system simulator. Based on a definition of a computer module detail and the system studies and data flow analysis accomplished a system simulator is developed. The simulator first includes the instruction logic simulation, later stages of the simulation include the entire multicomputer simulation capability. Two objectives are accomplished. First, it proves out the system design and, secondly, a capability is provided for the future systems application analysis.

7) System application analysis. With the help of the system simulator which has been developed and with inputs from the continuing operational requirements which are under development, the computer system is instantly analyzed as to how it will be applied and how it meets the known requirements. This includes the detailed programming of many applications to obtain timing information and memory requirements. Much of this analysis can be accomplished on the prototype equipments at this stage of the development.

8) Specify ancillary and peripheral equipment. As a result of inputs from the application analysis it is now possible to specify what ancillary and peripheral equipments will be necessary to round out the total system capability.

9) Field test and system refinement. At this point the first prototype equipments are on hand and have been extensively programmed to accomplish certain tests under field conditions. Some of the required tests may take place in conjunction with simulation of the system characteristics on a general purpose non-military computer such as was done with CCIS-70. As a result of this testing and further analysis, system refinements are spelled out and all information which has a bearing on the procurement of fieldable equipment is developed and collected.

10) Specify amounts and types of equipment. This is the collection of all of the information into a request for a quotation on the basis of which fieldable equipments will be procured in the amounts and types desired.

# Appendix A

# TWO ADVANCED GENERAL PURPOSE DIGITAL COMPUTERS

## THE LITTON L-300 SERIES COMPUTERS

The Litton L-300 family of computers is a series of integrated circuit militarized computers, currently consisting of the L-304, L-304A, L-305, and L-306. These computers are very similar logically and programs are interchangeable between the various models. They differ primarily in speed characteristics and in certain specialized features offered for the larger machines.

This computer is intended to fulfill requirements for military needs such as for ATDS and MTDS applications and is described as offering the capability of a large general-purpose digital computer in micro-miniature packaging. It is the contention of the manufacturer that computers of this type may serve in many different capacities. For example, although it is designed primarily for use in a field shelter environment, it is also considered as well fitted for shipboard use and as an appropriate component for a conventional data processing operation. It is also sufficiently miniaturized to be considered for avionics applications. This attitude among manufacturers is important to note and will become a more compelling argument as the cost of miniaturization decreases relative to standard componentry.

### Computer Characteristics

The characteristics of the L-300 computers are summarized as follows:

Physical Characteristics

| | |
|---|---|
| Volume | 0.3 cu ft to 1.1 cu ft * |
| Weight | 27 lb to 86 lb * |
| Temp | $-55^{\circ}$ to $+125^{\circ}$C |

| | |
|---|---|
| Physical Characteristics (Cont.) | |
| MTBF | 2300-8500 hours depending on configuration |
| Cooling | Fans on outside of case for air circulation |
| Power Consumption | 140w-430w |
| Word Length | 32-bit instruction words, 16 or 32 bit data words |
| Speed | 1.92 μsec read-write cycle; 120 nanosecond clock interval |
| Memory | 4096 words expandable to 32,768 words |
| Instruction Repertoire | 62 instructions including special data handling commands |
| Multiprogramming | Program Levels |
| Input/Output | Up to 84 I/O devices |

*Depending on model and memory size

A comparison of the capabilities and features of the L-300 computers is indicated below:

| | L-304* | L-305 | L-306 |
|---|---|---|---|
| Execution Times | | | |
| Add | 7 μsec | 4 μsec | 2 μsec |
| Multiply | 28-38 μsec | 11 μsec | 9 μsec |
| Special Features | | | |
| Multipurpose Process Registers | X | X | X |
| Scratchpad Register Memory (0.3 μsec) | | X | X |
| Look-Ahead Control | | | X |
| Simultaneously Accessible Memory Modules | | | X |

*The L-304A is similar to the L-304, an additional feature is the addition of 32-bit arithmetic.

## Multipurpose Process Registers

The L-300 computer programs utilize a set of 8 multipurpose 16-bit registers, each of which may be used as index registers, as accumulation, and for shifting and logical operations. One of the registers doubles as an instruction counter. For some operations two adjacent registers may be coupled to provide 32-bit capacity.

These registers are similar to the general registers in the IBM System/360. The use of this type of internal organization may be identified as an industry trend. It permits the programmer access to the machine at a low logical level without imposing unusual complexity. The multifunction aspect of general registers provides flexibility in addressing and increases the manipulative inventory of the computer organization.

A notable feature of the Litton design is the provision for 64 sets of the multiple purpose registers to facilitate multiprogramming. Thus, each program has its own set of general registers which remain intact in case of interrupt.

The mechanization of the multiple process registers varies according to model. For the L-304 the registers are part of the memory — the low-order 512 cells. For the L-305 and L-306, the multiple process registers are contained in a special 0.3 μsec scratchpad memory. This is in line with the trend in computer design to make distinct the logical structure versus the physical structure of the machine. The logical structure can then be kept consistent throughout a computer family to provide upward and downward compatibility and program interchangeability. The physical structure may be varied to provide economic versus performance trade off feasibility. This philosophy is especially espoused by the IBM System/360 and is adapted in the Litton computer family concept.

## The Instruction Format and Order Code

The instruction format includes a function field and two 3-bit fields which specify which process register is to be used as an address, and index register as shown below. It also includes a 3-bit address option (M) which can specify any of 7 options. Indirect addressing, indexed addressing and literal addressing may be specified in various combinations. The last sixteen bits (A) are used as an address field which allow addressing of 65K half-words; this field may also be used as a literal, that is, the field may contain the actual operand rather than the address of the operand. The left-most bit of the word (E) is an indicator used to cause a program trap in case of illegal arithmetic operations.

| E | F | H | M | S | A |
|---|---|---|---|---|---|
| 1 | 6 | 3 | 3 | 3 | 16 bits |

The instruction repertoire contains a number of special instructions designed for data handling, rapid field manipulation and limit testing.

The MOVE and MOVE AND ZERO instructions move a specified field from one register to a field of equal length in another register. (The two registers may be the same.) The obvious application for this command is message format conversion, which is a very prominent requirement for MTDS.

The four GATED COMPARISON instructions are used for limit testing and would be particularly useful in telemetry applications. The four commands provide for JUMP IF INSIDE, JUMP IF OUTSIDE, JUMP IF OUTSIDE AND GREATER, and JUMP IF OUTSIDE AND LESS.

It is noted that the special commands are hardware implemented, and do not constitute a combination of simpler commands as might be suspected. The ones named above all require the same execution time as the ADD command (7, 4, and 2 μsec for the L-304, L-305 and L-306 respectively); the MOVE commands require an additional μsec per bit that the field is shifted.

The instruction set also includes a NORMALIZE instruction, in case a floating point package is later needed, (useful also in interrupt processing), and an EXECUTE command. A good set of logic commands (both exclusive and inclusive OR) is also included.

## Addressing

The addressing modes for the Litton machines are unusually flexible. The available options are:

1) Direct
2) Direct with Indexing
3) Literal
4) Literal with Indexing
5) Indirect
6) Indirect with Direct Indexing
7) Indirect with Indirect Indexing

Memory is addressable in 16-bit half-words; however, each access consists of two consecutive half-words. A bit in the instruction word determines which half-word is to be accessed.

## Packaging

The modules of the L-300 series computers are arranged in drawers. The central computing section is contained in two drawers, the I/O section is contained in one drawer, and each 4096 word module of memory requires one drawer. The power supply units, which are each self-contained, also occupy one drawer (4 units). Three power supply units are required to operate the L-304.

## Systems Organization

Multiprocessing. Multiprocessing is a permanent feature of the Litton machines. Communication is achieved by two methods: 1) via shared memories, and 2) by use of the input-output channels. For computers which are colocated, bulk data transfer between computers is obviated by the use of shared memories. Therefore, only status and command information need be transmitted by I/O channels.

Memory sharing is achieved by means of a device called the Memory Allocator. This device allows any of up to 8 computers to access memory modules selectively for up to 262K of memory. The computers may block each other out and similarly, they may unblock access. There is no external switch override, however. A fixed priority system resolves conflicts when more than one computer attempts to access a memory module. The priority system also incorporates the Input/Output associated with each computer system. The input/output devices are each assigned a priority in respect to the memory modules which, is higher than the priority assigned to the computers. Program protection is provided by means of limit registers associated with each computer in the system.

Communication between computers may also be affected by data transfers on any of 64 input/output channels. This may be accomplished by establishing a master-slave relationship. In this case only one I/O channel is used and the data transfer is in one direction only. Alternately, with the use of two channels, transfers of data in either direction may be made. Communication between computers which are either colocated or remote may be effected via I/O channels.

Provision is made for the possibility of more than one computer having access to the same I/O device. In this case a fixed priority system determines which computer has control at any given time.

Input/Output. The basic I/O transfer rate for an 8-bit character or a 32-bit word is 4 μsec. The I/O operations may be initiated either from the program or by the I/O device. A time-sharing multiplexor is available which allows as many as eight I/O devices to share the same channel. Since the computer can handle 64 program levels, it is possible to relate each of the 64 devices to a program level and thereby determine I/O priority by program setting of a priority table.

Interrupt logic depends on the setting of bits in two tables of control bits composed of 64 bits each of which contain status bits and interrupt-enable bits. These tables together constitute what is known as the Program Activity Register (PAR) respectively. A program is eligible for operation if both bits relating to its program level are set. However, such a program will operate at a given instant only if it has the currently highest assigned priority. Thus, to be recognized, an interrupt must be assigned a priority higher than the currently operating program, otherwise response to the interrupt will be deferred.

## Programming and Software

Multiprogramming. The computer is especially designed to accommodate multiprogramming. Sixty-four program levels are provided and each program has its own set of general registers. Program priority may be program determined by the setting of a priority table. The mechanism for determining which program operates is the Program Activity Register described in the foregoing section.

No hardware protect feature is provided for the L-304. However, in the L-305 and L-306, provision is made for program floatability. In the L-306, the assumption is made that memory protection is performed via software (as a part of the executive function) together with limit registers associated with each program level.

The use of the multilevel programming concept facilitates the common use of general subroutines and obviates the need for complex programming to solve the reentrant problem which would otherwise be required.

Multiprogramming is anticipated for three distinct environmental situations. These are:

1) Program mixes are completely defined and not subject to change. This situation does not require a distinct executive program.
2) Program mix changes periodically but predictably. A limited executive program is required for program loading.
3) Program mixes change dynamically and vary unpredictably. A real-time executive for scheduling and facility allocation is required.

The first environment is anticipated at the lowest echelons and the others as successively higher ones. These environments also relate and have helped to determine the design of the Litton computer family concept, which specifies the L-304, L-305, and L-306, respectively, for the varying requirements.

Programming and Relocation. Accordingly, the L-305 and L-306 include the capability for program floatability. This is accomplished with the use of a base address register which is added to the location register to obtain the effective instruction address. Relocation of programs within memory is accomplished by changing the base address register.

This method works in the case of instruction accessing but may cause difficulty in the addressing of data depending on whether the addressing is relative (to the program) or absolute. Certain programming restrictions are therefore imposed in order to preserve program compatability between machines.

Software Support. The programs developed, or to be developed include an assembler program, (CS-1), and an operational system composed of standard utility programs. A 7094 Logic Simulator Program was also written to test the L-304 internal logic.

Status

An engineering prototype L-304 with 4K memory is in system checkout. The logic units have been checked out and the memory unit is now being integrated.

In this test system the I/O Console, Control Console, Display Console and Magnetic (loop) Tape units have been tied in.

Litton is currently writing proposals incorporating this equipment on a fixed price basis.

## THE AUTONETICS COMPUTER SYSTEMS

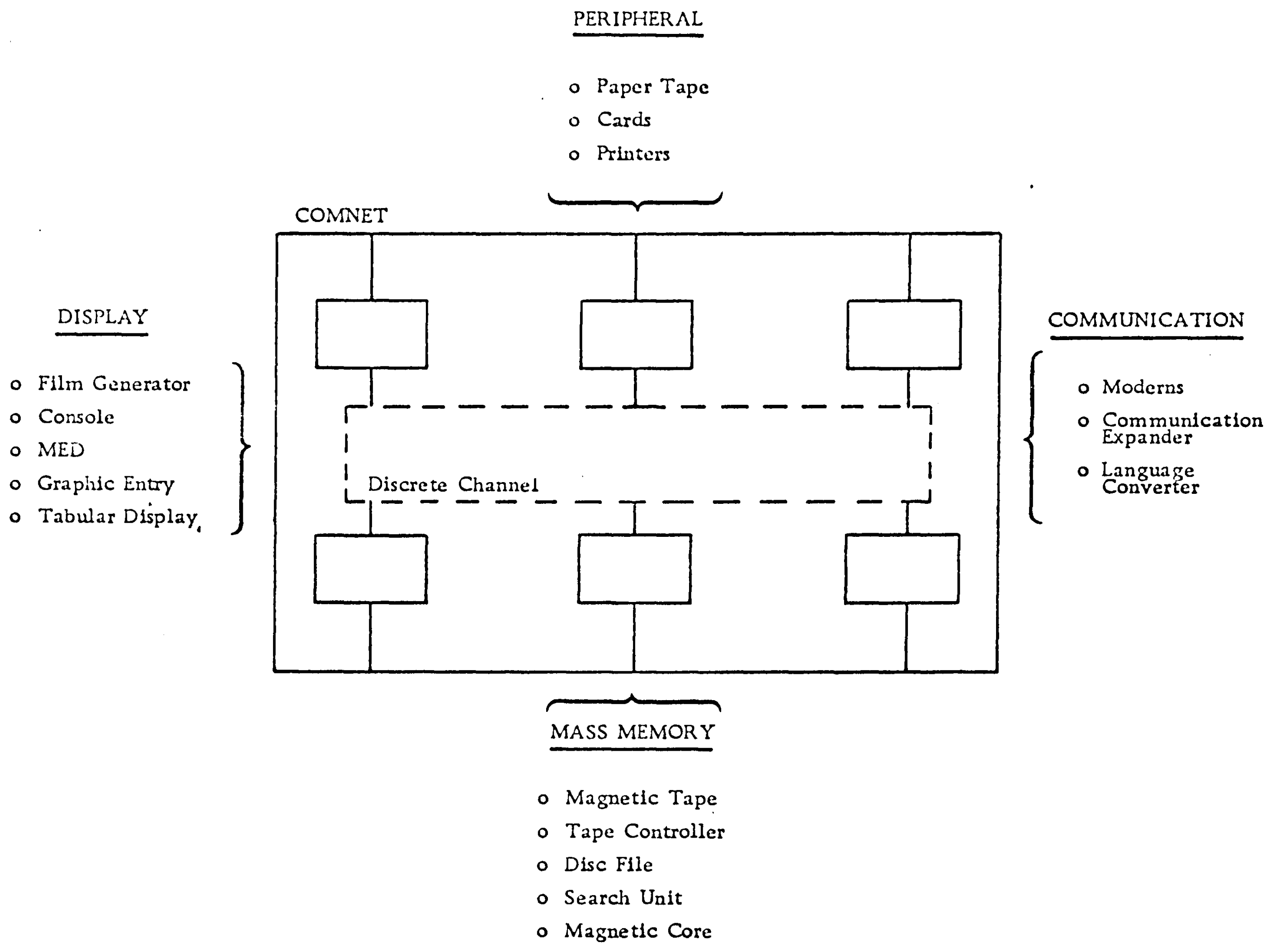
The Autonetics Data System Division of North American Aviation, Inc. has figured prominently in the development of miniaturized computers for military use. Among these developments are the Monica family of computers, the D26C and D26J, and the MINUTEMAN computers, the D17 and D37B. In addition, Autonetics has developed system concepts which are based on studies of CCIS-70, 492L, 48 1L, 407L, and other systems. Of particular interest are a number of more advanced versions of their militarized computers which are proposed for the ground mobile environment in advanced military systems. Among these are the D-28C, the D57, and the D58.

### System Concept

A primary characteristic of the Autonetics system philosophy is the incorporation of a multiple computer system (MCS). The MCS is highly modular and may be arranged in several alternate configurations and operational modes suitable for various echelon levels and for varying requirements. The system is characterized as a multiple-computer distributed-function system.

Computation capability is increased by adding computer modules rather than by providing unusual speed capability. This system uses the "unit computer concept" rather than the "family of computers" concept. Autonetics has been studying several generic systems to apply to specific tactical requirements.

The system concept may be described in terms of a chart, as shown in Figure A-1, which illustrates six computer modules. More or less (even one) modules can be used in a system.



FEATURES

- The disruption of communication at any one point will not destroy capability.
- Discrete channel used for command information and exchange of status information.
- COMNET used for data transfer (14 channels).
- No dedicated equipment.

Figure A-1. Autonetics System Concept

## The COMNET

The system communication structure is based on a decentralized switching scheme called the Communication Network (COMNET). The COMNET consists of a number of bi-directional channels used for communication to high speed and low speed peripheral equipment and for "real-time" communication to communication terminals. A typical configuration including 14 channels is shown in Figure A-2. Also shown is the Discrete Command Channel, not a part of the COMNET, which is used for communication of command and status information. It consists of a single path for the use of command requests (from human intervention) and for control functions exercised from the executive computer.

Each of the real-time channels may accommodate 6 devices via a multiplexor device called the communication expander. This device will provide terminals for 6 communications devices. Thus, a single module can be tied to up to 36 communications devices.

Data traffic carried by the COMNET channels is classified as programmed or nonprogrammed depending on whether the transfer is under program control or externally initiated (e.g., by operators, remote data terminals, another computer). This distinction corresponds to the usual categorization of interrupts — internal and external.

## Autonetics Computer Characteristics

The characteristics of the D28C and the D57 computers are summarized as follows:

| | D28C | D57 |
|---|---|---|
| Physical Characteristics | | |
| Volume | 3 cu ft | 3 cu ft |
| Weight | 100 lb* | 140 lb |
| Power Consumption | 585 w | 1100 w |
| Packaging | Field Case | Rack or field case |

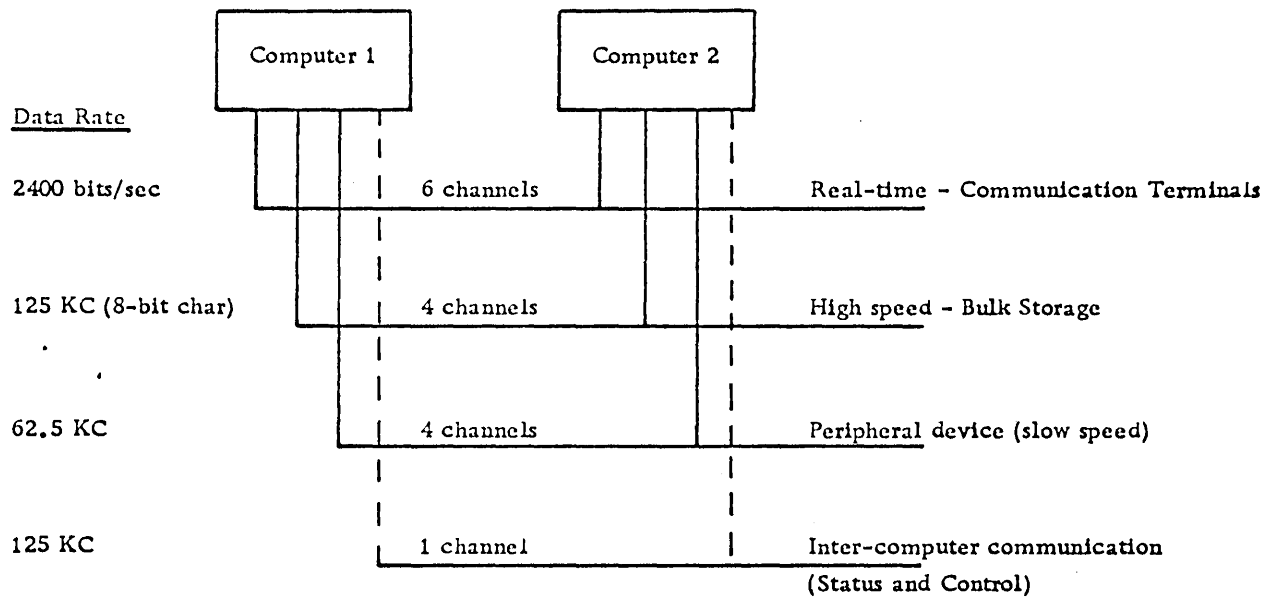


Figure A-2. Communications Network

| | D28C | D57 |
|---|---|---|
| Word Length | 36 bits | 36 bits |
| Speed | | |
| Cycle | 6 μsec | 4 μsec |
| Typical Operation Times | | |
| Add | 12 μsec | 8 μsec |
| Multiply | 48 μsec | 44-48 μsec |
| Transfer | 6 μsec | 4 Usec |
| Memory | | |
| Internal | 2 modules of 4096 words ea | 2 modules of 8192 words ea |
| Internal Memory Extender | Up to 6 modules of 4096 words ea | Up to 2 modules of 8192 words ea |
| Instructions | | |
| No. of instructions | 59 | 68 |
| Special Instructions | Half-word manipulation repeat, move | Bit and byte manipulation, repeat, move DCB, BCD, and floating point |
| Addressing | Immediate (literal) Direct and Indirect | Immediate, Direct, and Indirect |

*Includes control panel and field case

Input/Output System

The Input/Output system features independent and simultaneous operation of I/O and computing operations. This is achieved through the use of a second memory module which may be independently assigned to the I/O unit. A multiple stacking interrupt system provides for asynchronous input of data.

The communication scheme permits each processor to communicate with each device in the system. Alternate paths between processors and devices are permissible and at least two paths exist between critical units. The communication switch nodes are processed by each processor internally (in the I/O unit) so that failure in the switching elements effects only that device. This scheme avoids the characteristic vulnerability difficulty in multicomputer systems which depend on an external switching unit. The physically decentralized switching system therefore promotes high system availability.

Intercomputer communication may be achieved via the Discrete Command Channel for command and status information or by the I/O unit'channels for data transfer. Initially the design called for special intercomputer channels. However, subsequent analysis using the Gordon Simulator Program suggested that the discrete channel and the normal I/O channels were adequate for the purpose.

The Input/Output System is Fieldata compatible and, with minor modification, is also compatible with ASCII requirements.

## The Computer Organization

The computer organization of the D-28C and D-57 are similar and may be thought of as variations of the same design. They differ only in minor respects, such as the size of the memory banks and the provision for additional memory modules. This organization features two memory banks and provides for simultanity of processing and I/O. The computer internal data/control flow is indicated in Figure A-3

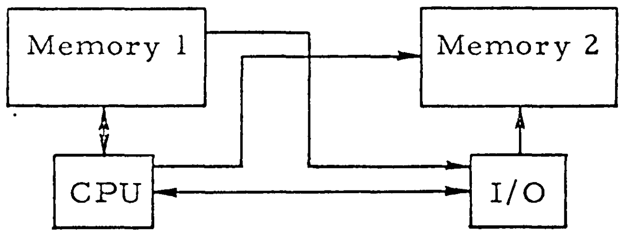


Figure A-3. Computer Internal Data/Control Flow

An advantage of the multi-memory arrangement is the provision for the overlapping of the memories in order to achieve effective speed increases. This is achieved by a look ahead feature which interleaves instructions (stored in one memory module) and operands (stored in the other).

## The Memory System

In addition to the two interval memory modules additional memory modules external to the computer may be used to increase storage capacity. These modules may be individually addressed by either the I/O unit or the processor unit. Each D-28C computer can operate with a maximum of eight 4K memory modules, 2 internal units, and 6 external ones. With this arrangement it is possible for the processor unit and the I/O unit each to operate with four memory modules simultaneously. The D57 computer can operate with 2 internal and 2 external 8 K memory modules.

The external memory is called an Internal Memory Extender (IME). An IME of 24 K capacity weighs 94.5 lbs, requires 500 watts of input power and occupies 3.6 cu ft.

In a multiple computer system the IME's are ordinarily shared between the system computers and are not dedicated to any one computer or I/O unit. Communication between the computers and the IME is effected by means of a logic switch.

## The D-28 Interrupt System

A multiple stacking interrupt system permits asynchronous input of data from external devices. Interrupts may result from events external to the computer or may be program initiated (internal interrupts). The interrupt system is affected by both hardware and software features. An executive interrupt routine performs the branching control for real-time operations. The hardware features include the usual provision for saving and restoring the necessary working resistors. A mask register provides the mechanism for selectively inhibiting interrupts under program control.

The interrupt structure of the D-28 computer is indicated by the following lists of interrupt types.

Internal

- Status
- Program
- I/O Initiation and Completion
- Real-Time Clock
- Inhibit
- Diagnostic

Discrete Channel and I/O Channels (External)

- Status
- Malfunction
- Count Down
- Action Request and Completion

## The Executive Concept

The Executive Program has three levels; 1) Input/Output control routines, 2) Program sequencing and control, and 3) System Configuration Control. Each computer in the multiple-computer system would have the first two levels of executive control. However, only one computer (at any one time) would have the third one. This computer would therefore be the master computer. As a part of the System Configuration Control function, the executive maintains a table of system hardware elements which it updates according to current assignment and scheduling. For example, when Internal Memory Extenders are used in a multiple computer system, they are assigned temporarily to particular processing units by order of the Executive Computer. When the assignment is completed the memory is available for reassignment by electrically switching the memory to another unit. A design objective of the Executive Control Program is to permit the user programmer to write programs without having to make allowances for how many computers or peripheral equipment are present in the system.

A feature which enhances processing availability is the presence in the system of an on-line spare. In case of failure of one of the other computers, this computer would be available for the immediate resumption of the functions of the failing computer. The on-line spare would normally receive data from the other computers to maintain currency of the essential data base. It is also suggested that this computer could be assigned low priority tasks as well. Self-checking functions are performed in each computer, and overall system health checking is performed by the master executive and monitored by the computer operator. In case of executive failure a restart capability is provided.

# Appendix B

## GENERAL MULTIPURPOSE REGISTERS

The use of general rather than specialized registers in computer organization is an important development. These registers are used for a multiplicity of purposes. For example, the multipurpose registers of the Litton 300 series computers contains operands and addresses; they may assume the function of accumulators, shift registers, index registers and program counters; and they have the associated circuitry for the performance of arithmetic and logical operations. Adjacent registers may be coupled to provide double word length shifting and arithmetic operation. The general registers are addressed (referenced) by a three-bit field contained in the instruction formats. The technique of using general multipurpose (usually interchangeable) registers has reached its most advanced development in the design of the IBM System/360. In this family of computers the mechanization of the general registers varies from model to model; however, the programmer does not need to be aware of the differences, since, logically he is presented with the same general register capability in each model. Earlier examples of this technique are found in the stored logic microprogrammed computers, the C-8401, the Raytheon 440 and (to a lesser extent) the BR-130 (AN/UYK-1).

This design obtains a natural appeal to the programmer who is given an inventory of manipulatable elements that can be used interchangeably in various ways. He is provided many receptacles for his interim results (many accumulators), can use a value as an operand at one step, an index value at the next. In effect, these registers provide a local, fast-access storage which permits operations involving several operands to proceed without memory access delays.

Apart from individual programmer preference, the use of general registers has a more compelling rationale which is reinforced by the nature of the application span of tactical command and control problems. This is understood if one recalls the traditional and current gap between memory speeds and live register speeds. To narrow this gap dual memory designs, fast control memories and scratchpad memories have been employed. However, to a degree, the logic designer has been content to provide a faster capability for operations than for operand transfer. The register speed is a determinant for operator speed, that memory speed to a large extent is the determining limitation of operand transfer speeds. This speed gap is appropriate when the main computer function is primarily a sequence of calculation. However, for the tactical command and control application, the greatest importance attaches to transfers of data, conversion of data, and in particular, to storage and retrieval of data.

The conclusion to be derived from this discussion is the relative importance of memory access speeds, and insofar as the disparity between memory access and live logical register operations still exists, the faster speeds of fast scratchpads and general registers should be used for, and geared toward efficient data conversion, retrieval and transfer.

# Appendix C

# MEMORY DISCUSSION

## MEMORY HIERARCHIES

A memory hierarchy is a set of memory elements having various levels of speed and capacity. The speed and the costs per bit of the various memory elements usually vary inversely with the size (capacity). The ideal hierarchy is usually described as one with a fine gradation of speed and capacity characteristics, from small amounts of very high speed storage (registers and scratchpads) through high speed memory main storage, to successively larger amounts of lower speed storage.

The attention to memories and their hierarchical organization is of importance because in future operations common accessibility to data bases and the methods of organization of information in those data bases so that easy, rapid, and accurate recall of such information will be possible is a cardinal problem in the design of computer based or computer aided systems. Where the rapid retrieval of information is necessary, in such applications as message switching or intelligence file retrieval, the particular way in which the different kinds of memory are organized with respect to each other may be vital. Especially is this the case in which the available times of response of the computer based system is of the order of milliseconds or shorter.

Although memory hierarchies are characterized mostly by speed and size, other aspects are noteworthy. Specialization of memory functions, topological relationships of memories with processors, and control hierarchy are also important topics for consideration. In particular, the development of read-only memories, read-mostly memories, and content-addressable memories is important to notice.

The advantage of a multi-level hierarchy is to increase the effective speed of the computer by placing the programs and data of most frequent usage in the highest speed devices and, at the same time, realize the economies of less expensive storage devices for data not subject to rapid change. Techniques have been and are being developed for exchanging data between memories on a dynamic basis. Transfers are made between scratchpads and main memories and, on a slower time scale, from main memory to bulk storage.

## CONTENT-ADDRESSABLE (ASSOCIATIVE) MEMORIES

The use of associative memories as possible components for future tactical systems appears promising. Several application areas of interest for which associative memories are candidate are:

1) Logistics File Maintenance
2) Intelligence Retrieval
3) Radar Data - Track Data Correlation
4) Scan to Scan Radar Data Correlation
5) Track Data Retrieval
6) Weapons Assignment
7) Sorting
8) File Search for Communication and Display
9) Message Format Conversion
10) ECCM
11) Decoy Discrimination

## CONTENT-ADDRESSABLE CHARACTERISTICS

The addressing technique employed by conventional memory organization requires that the location of the information to be retrieved be known. This is sometimes referred to as coordinate addressing or addressing by location. An alternative addressing technique is the retrieval of stored information on the basis of content. Word cells are accessed by the characteristics of the stored data rather than by the physical location of the cell. This eliminates current very tedious indexing operations. The physical location is immaterial in this case. Memories with this capability are said to be content-addressable and are also referred to as associative memories, parallel search memories, recognition memories, and several other terms.

The term associative memory has only recently come to refer to a particular hardware design. Formerly, the term was used to refer to the use of conventional memory for purposes of associative logical processes. This usage of the term persists and is of continuing validity.

The development of new addressing techniques is not without precedent. It is sometimes suggested that the development of contents addressing is analogous to that of indirect or other recognized addressing techniques (e.g., relative, indexed, implicit, immediate, truncated, etc.). To a similar (or greater) extent, contents addressing provides generality to the common task of "operand fetching."

The problems of addressing are taken for granted to such an extent by the programmer that it is perhaps not evident that the association of an address with a quantity of information is usually a somewhat superficial relationship. Unfortunately, the programmer, particularly, the machine language programmer, must often deal largely with addresses rather than information itself. This preoccupation with location imposes a large housekeeping and data organization requirement which can be expensive of machine execution time and storage, and programming effort. Although it is perhaps not unreasonable to expect the programmer to keep track of and in some cases plan where information is to be kept, it is interesting to conjecture concerning the advantages to be obtained by a programming technique substantially free from the addressing requirement. This is feasible with several designs of contents addressable memories. Information is simply stored in the first available cell and may be retrieved later in one cycle (or two) without knowledge of its physical whereabouts.

## ASSOCIATIVE MEMORY CAPABILITIES

Associative memories currently under consideration compose a broad spectrum of capability — from the simplest search based on the equality criterion only, to searches using selected bit patterns (masks) and based on any of perhaps a dozen search criteria, and extending to proposals which include computation on matched items. The feature in common with most designs is the capacity to interrogate simultaneously, a region of the associative memory based on comparison with an external key. The characteristic property is economical search speed. In the more advanced designs, the individual memory cells are capable of elementary logical decisions in relating the stored data to the search criteria. By a modest extension of the logical capability, parallel operation of conventional commands of the simpler sort are obtained.

## SEARCH CRITERIA

Search criteria are presented simultaneously to all or selected subsets of the associative memory. Criteria other than identity with the externally presented key are possible. For example, the associative memory may be designed to search on the basis of--greater than, less than, between limits, minimum, maximum, next higher, next lower, etc. In addition, mixed modes may be specified involving combinations of independent searches performed in sequence. This is also referred to as a successive search criterion. In addition, a simultaneous search criterion may be provided which accomplishes a search on multiple criteria in one search cycle. Composite searches in which different search criteria are used for each of several fields in the search word are also proposed in some designs. The "m of n" logical relationship is also possible; for this process, a successful search is defined as one which satisfies the m out of n criteria (e.g., any 3 out of 4 specified characteristics).

The satisfaction of the specified search criterion is said to result in a "match." A search will result in a match, no match, or in some cases multiple matches. Some designs include the option of specifying that the search be terminated after the first match or after n matches occur, where n is a preset threshold parameter. A more usual method is to provide a multi-response resolver which provides unique to non-unique processing to allow a serial treatment of matched items. Typically, the matched cells are identified, i.e., tagged or marked, in order that the contents may subsequently be retrieved.

## DATA CORRELATION

A general classification of computer problem is data correlation. An example of this type of problem is the requirement in many high track capacity surveillance and fire control radar systems to correlate radar (sonar) data with established tracks. It is customary in such systems to keep account of the currently known trackable entities in the computer store. These entities, once recognized, become known as "tracks." The information of interest concerning each track is usually kept as a body of information (as a block of track data) and dynamically updated by current inputs. Examples of track characteristics are track position, track velocity, track identification, threat priority, predicted position, etc.

The association of a set of incoming radar returns with existing tracks is called radar correlation. This function is performed by determining according to a tracking correlation logic if a given return lies within a volume of space which is centered about a predicted track position. If it does, it is said to correlate.

With a conventional computer the correlation process is quite time consuming since each return must be compared with all track positions. The processing time, therefore, increases approximately as the square of the number of current tracks. With a parallel search memory, however, it is possible to compare the position of each return with all track positions in one search time. The track correlation problem solution time thus increases only linearly with the number of tracks. The process is particularly efficient in an associative memory processing between limits search capabilities.

## TRACK DATA UPDATING AND RETRIEVAL

In the general application areas of surveillance and tracking other associative memory usages are apparent. In such applications, it is often required that track data be updated frequently from data received externally, and it is frequently found necessary to retrieve track information by specifying one or more of the track characteristics. In a conventional memory, the ordering of track data is according to one or more of these characteristics (e.g., track number, track channel number). Sometimes data concerning the same track is kept in different physical locations and ordered differently as may be required. This will ordinarily necessitate that special cross referencing items be maintained. If the ordering criterion is not known (as in the radar data correlation problem), a search is required entailing observation of each item. With the use of an associative memory, track data could be retrieved by specifying any of its known attributes.

## SORTING

Sorting involves the sequencing or resequencing of data such that the newly formed sequence satisfies some specified ordering relation. Sorting has become a major computer application and one which is quite (computer) time consuming. The importance of the problem is evident from the emphasis placed on research into efficient sorting methods. Sort programs of considerable length and complexity are written and improvement of method is continuously sought.

There are several aspects of the sorting problem which are related to the development of associative memories. The most obvious advantage for such a usage is that in many cases the need for sorting is eliminated. Since the data may be addressed by its known characteristics, the need for ordering it is often obviated.

However, the need for sorting would still be required for certain tasks: for example, when sorted data must be transmitted to some other memory device, for preparation of sorted output lists, etc. If sorting is necessary, the associative memory may efficiently be put to the task.

## USE OF ASSOCIATIVE MEMORY FOR SYSTEM PROGRAMMING

The production of system support programs such as assemblers, compilers, translators, etc. is an appropriate application area for associative memories. In such programs, the functions of scanning, table building, table lookup and table search are very common ones and in fact constitute a large part of the computer activity. In a machine featuring an associative memory, these functions are greatly facilitated. Tables used during the assembly process could be stored in the associative memory to improve search speeds.

Another aspect of assembly and compilers program is relocation of programs and data, and the assignment of absolute locations. Contents addressing eliminates the need for some of this activity since data can be entered without undue concern about storage allocation. For example, it is not necessary to determine in advance if contiguous areas of core storage of the appropriate size are available nor even keep track of data locations if retrieval keys are stored with the data. The need for translation from symbolic to absolute storage location may be to some extent eliminated. In a sense, the cells could contain their own names (simply another attribute — an array of bits which could be used as a basis for search).

## APPLICATIONS

Several applications can be cited where the use of an associative memory can result in order of magnitude time savings in the solution of problems particularly amenable to solutions using such a memory. Other applications show little or no improvement. Each application therefore must be considered independently.

The applicability of associative memories to future tactical data systems depends largely on whether significant portions of the processing problems are of a type for which the content-addressing capability is demonstrated to possess decided advantages over conventional (location) addressing. It is, therefore, important to determine those problem characteristics which tend to guarantee that problem solution time will be decreased (or effective work increased) with the use of this capability; and to define classes of data processing operations which make effective use of the inherent parallelism afforded by associative memories.

Content-addressing is efficient relative to location addressing for those applications for which:

1) Files of data are manipulated and search for individual items in the data store is a significant part of the data processing task.
2) Fast or immediate access to data is needed in order to avoid table search functions.
3) Data is retrieved on the basis of a multiplicity of reference properties and cross referencing between files is a frequent program activity.
4) List organized data storage is required.
5) Sorting is performed.
6) Encyclopedic data is stored from which retrieval of individual items is called for.
7) Cataloging and cross indexing is required.
8) Comparisons with data arguments are needed quickly and test answers concerning item characteristics required.
9) It is necessary to order data quickly for output or in the case where data tends to become disordered dynamically during processing, or must be reordered according to varying criteria.
10) Problem solution is obtained by solving large sparce matrices.

These characteristics are to a large extent, concentrated on data-oriented problems. In particular, data with the following attributes:

1) Data with multiple arguments.
2) Data for which unpredictable growth of tables, arrays, or other structures appear likely.
3) Data with a large proportion of void or zero elements.

Retrieval execution time is largely independent of table length. The associative memory advantage is, therefore, more pronounced as file size increases.

Several applications which are anticipated for future tactical data systems are of the type indicated in the foregoing list of characteristics. Several are discussed briefly in subsections to follow. Beyond the special applications for which advantages are easily seen, lie new usages derived from new software techniques. These further advantages will accrue from the reformulation of problems from the standpoint of explicitly exploiting the inherent parallelism; and from the unknown but predictable advances to be derived from experimentation in these techniques when associative memories are a practical hardware reality.

## ADDITIONAL APPLICATIONS

The associative memory would be useful for file searching of data for display, communication, and data updating. Other applicable areas of interest are ECCM decoy discrimination, threat evaluation, and general areas such as:

Reservation System
Air Traffic Control
Automated Intelligence
Automatic Abstracting
Matrix Arithmetic
Document Retrieval
Language Translation
Compiler Writing
Data Retrieval (Medical, Legal)

Payroll
Problem Solving
Perpetual Inventory
Linear Programming
Automated Teaching
Communications Switching
Airline Reservation
Process Control
Vehicle Registration
Mathematical Applications

Many of these have no foreseeable usage in connection with tactical systems. However, the fact that there is such a wide potential usage does relate to the question of whether such devices will be available in economically commercial versions. If the range of applicability is broad enough, the motivation for commercial development will be provided so that successful implementation will not depend entirely on subsidized development.

## RELEVANCE TO MTACC

There is a strong likelihood that many MTACC applications warrant the use of an associative memory. Because it is usually impossible to forecast all of the uses to which files will be put and because even present uses of files may change in nature, it is necessary to develop a technique of system design and to use subsystems that will be as independent as possible of particular detailed requirements. For example, an intelligence file contains information that is updated and retrieved in accordance with current well specified procedures. It is foreseeable that many questions about data in the file will be asked which are not amenable to the current file organization or to the ways in which information is currently retrieved. Though this information is resident in the file, there is a premium on not having to reorganize it to secure this information. It is also possible that other functional areas, such as ground combat operations will also wish to query the file for information in a manner that will not, with one form of file organization, be easy. Indeed, if the files for intelligence and ground combat operations are to be combined, then two different sets of criteria may have to be rationalized for the extraction of information. In general, this problem of working with unstructured files or with files that have previously been given the stamp of one form of organization is one in which the developments in content-addressable memories will find considerable application.

For MTACC, the use of associative memories at the higher echelons is indicated for the functional areas of intelligence, logistics, and air support.

The advantages offered by associative memories are those of speed and flexibility. Since it is possible to retrieve desired information at a rate which is independent of the amount of memory searched, associative memories can save substantial time in applications which involve a great deal of searching. There is also a case to be made for increased ease of programming, since the requirement for certain internal housekeeping functions concerned with problems of addressing are simplified. This is offset, however, by the necessity for using techniques which are currently unfamiliar to the programming community.

The drawbacks are largely those related to implementation difficulties and the attendant costs associated with development and production. The cost for workable associative memories is much higher than for conventional memories of equivalent size. However, the important cost consideration is that of total system cost versus overall system capability. It is in this context that the utility and economy of associative memories will ultimately be weighed.

## READ-ONLY AND READ-MOSTLY MEMORIES

An important type of memory which is finding increasing application in computer systems organization is the read-only memory (ROM).

The cost of read-only memory is considerably less than for read-write memory of equivalent speed. This type of memory is referred to as permanent, since the machine using it is ordinarily not capable of changing its contents. Often, the program which does not change in use, constants, tables, and coefficients can be shared in the less expensive ROM with a separate read-write memory being used for that data which changes.

Read-mostly memories whose characteristic feature is a fast-read, slow-write capability are also coming into increasing use. Here again, the criteria for the development is a reduction in cost. This allows data that seldom changes to be updated inexpensively without impairing the ability to retrieve the data rapidly.

A number of important usages are envisioned for read-only and read-mostly memories; among these are:

1) storage of micro-programs (stored logic)
2) storage of arithmetic tables
3) multiprogram control
4) function generators
5) storage of executive control programs
6) code conversion (e.g., radix conversion)
7) interrupt processing routine storage
8) storage of test routines

Read-only and read-mostly memories are likely to be used in several MTACC computer systems. They may be used to substitute for a part of main memory or for faster access for large capacity storage at lower cost. The primary characteristic of programs which can effectively exploit the read-only memory speed, are those which are relatively immune to change and whose frequency of usage is sufficient to result in significant time saving.

# Appendix D

# COMPUTER LANGUAGES

## FORTRAN

A version or dialect of FORTRAN was one of the first languages for which a widely-used operational compiler was completed. It is important for several reasons. It is well suited to a broad range of scientific applications. FORTRAN processors have been written for a large number of different types of computers. The concepts it embodies have instigated much research in computer languages and many attempts to develop computer languages for non-scientific applications. The problems encountered in writing FORTRAN processors have prompted the development of most of today's compiler technology.

The capabilities of a particular FORTRAN processor are a function both of the machine for which it was written and the capability of those who wrote it. Processors for smaller computers have fewer facilities. The newer processors contain fewer program steps, work faster, and produce more efficient machine language programs. The extra facilities which are provided in some cases or the particular facilities omitted in the processors for the smaller machines depend on the inclinations and background of the writers.

The basic language is most powerful for applications requiring matrix manipulations. It is a very useful tool for most scientific applications. For the normal type of commercial problem it is rather inefficient, and caution should be exercised in using it on applications which are largely logical. But it would not be difficult to enrich a FORTRAN processor with functions and subroutines to the point that it would be useful for applications involving logic and be, in some cases, quite satisfactory for commercial data processing.

The principal drawback to FORTRAN is its vague and less than optimum syntax, and its tendency to contain a degree of dependence. Because of this, a FORTRAN program which operates properly after being translated by a given FORTRAN processor may not work properly after being translated by another FORTRAN processor which was written from the same specifications. Because the syntax is less than optimum, the processors are more complicated than necessary.

## COBOL

The COBOL language was developed by representatives of computer users, Government installations, and computer manufacturers at the instigation of the Department of Defense. The intent of the committee was to provide a problem-oriented programming language for computer users with business applications. The language was to be machine independent, amenable to further development, and easy to learn. The programs produced were to be self-documenting. Ideally, the problems of converting programs for use on different types or later models of computers would be eliminated and programs could be produced by personnel relatively inexperienced in the workings or language of computers.

The language is machine independent, but the programs it produces are machine dependent to a significant extent. Programs which run efficiently on one computer do not run efficiently on another. Some manufacturers have special added features to the language which encourage the programmer to write programs which are more efficient for their equipment, but cannot be run on competitive equipment. The language is officially amenable to development, but it is evident now that, although it is amenable to change, the basic concepts preclude the integration of new techniques in a reasonable manner. Ease of learning and self-documentation were achieved at the expense of enforced verbosity.

It can be predicted that the business data processing world will move rapidly away from symbolic assembly languages. Many will adopt COBOL, but many others may delay the move until they see if PL/1 will become the de facto standard.

## ALGOL

The word ALGOL is a contraction of international algorithmic language. As its name might imply, it is the product of an international committee of computer language experts. Quoting from the report which defines and describes the language, "This is a language suitable for expressing a large class of numerical processes in a form sufficiently concise for direct automatic translation into the language of programmed automatic computers." It has the following important characteristics:

1) It is a procedural language.
2) The syntax is both concise and precise. The form in which it has been presented will probably become the canonical form for any new languages that are developed. It has been given the special name "Backus (or Backus-Naur) Normal Form" after these members of the development committee who invented it.
3) In the United States, acceptance and implementation have been slow.
4) It is recognized as the publication language for programs, algorithms, and techniques of general interest. But no input-output equipment can handle its character set. To date, it has no official input-output facilities.

Several factors have precluded rapid acceptance and widespread implementation. The language requires the use of many uncommon symbols with special meanings which are useful only in the writing of ALGOL programs. Manufacturers have been reluctant to provide the hardware required to read and print the symbols directly. The language is sufficiently unique that programmers have been reluctant to invest the effort necessary to master it. Some of the features are sufficiently difficult to implement and would be used so rarely that only subsets of the language have been incorporated into processors. In short, it has been a language for the expert.

On the other hand, ALGOL, as an influence on future languages will assume increasing importance with the passage of time. Its generality and capabilities are as great or greater than the class of computers for which it is intended. Peripheral equipment which will accommodate larger character sets, at such a price that it can be dedicated to the programmer, will be made available. Above all, the facility it affords for the precise, concise, and unambiguous expression of procedures in a machine independent manner will make it the prototype of languages for the computer expert of the future.

## JOVIAL

JOVIAL is a procedure oriented language produced by the System Development Corporation. It incorporates many of the features of FORTRAN and ALGOL together with special features gained from the System Development Corporation's experience with the development of the SAGE air defense system.

Although the language is patterned after ALGOL, use is made of self-explanatory English words and ordinary algebraic notations in a manner similar to FORTRAN or COBOL. Unlike most FORTRAN and COBOL processors, JOVIAL processors permit the incorporation of assembly language instructions.

One of the most important features of the language is the COMPOOL concept which permits a programmer to refer to and manipulate COMPOOL data. The COMPOOL is a library for a large programming system which supplies the JOVIAL compiler with data description parameters. Thus, it is unnecessary for the individual programmer to be concerned about or, in most cases, even be aware of data formats. By changing the data descriptions in the COMPOOL, it is also possible to change the manner in which the data is manipulated in all the programs making up the system.

Machine independence is achieved by introducing an intermediate language. The JOVIAL processors incorporate a generator which translates the JOVIAL programs to the intermediate language and a translator which converts the intermediate language to the machine language of a particular computer.

Early in 1965 the U.S. Army's Automatic Data Field System, formerly known as CCIS-70, undertook a study of computer languages. The objective of the study was to furnish a basis for selecting a standard language for Army use. JOVIAL was selected as the interim language for use by the Automatic Data Field System organization.

## CS-1

CS-1 is the language currently being used by the Navy in the preparation of programs for NTDS. This language provides some, but not all the facilities found in FORTRAN. Provision is made for control and arithmetic statements but not for input/output statements. The level of sophistication of control and arithmetic statements is not as high as in FORTRAN. Since it does not include input and output statements, the file definition facilities of COBOL are not present. The syntax is very simple and uncomplicated when compared to ALGOL, and is comparatively easy to use.

In general, only those functions necessary to the NTDS application have been provided.

## NELIAC

NELIAC (Naval Electronics Laboratory International Algebraic Compiler) is a procedure oriented language and is sometimes described as a dialect of the ALGOL family. It was developed concurrently with ALGOL 58, a predecessor of the current publication ALGOL.

It is an effective compiler from the standpoint of object program efficiency. Special facilities include partial word (and single bit processing, boolean operations and the option of including machine coded routines.

An interesting feature of the compiler program is that it is written in its own language. There are two major effects. It is possible to expand the compiler to accommodate any special features which may be desirable: it is self compiling. And, a compiler for one computer may be written and compiled on a second computer: it is therefore machine independent.

Next, a decompiler for NELIAC has been written. A decompiler is capable of producing symbolic programs from machine language programs. Through the use of a decompiler on one machine and a compiler on another, direct translation of programs may be achieved.

NELIAC compilation features the use of Current Operator/Next Operator (CO/NO) linkage tables. This method is based on the treatment of two consecutive operators and the intervening operand as a basic unit. During compilation, the program will transfer control to a particular program "generator" which will process the current CO/NO combination.

PL/1

PL/1 is the new programming language from IBM, the programming language currently being implemented for IBM's System/360. Although it is not radically new, it must be included because of the importance it is expected to have in the near future.

It is anticipated that the language will be extensively used. IBM intends to market System/360 to all its customers as a single product line and PL/1 will be the primary and best-supported programming language offered. Other manufacturers intend to market "compatible equipment" with compatible software. Probably, some of them already are implementing PL/1 for their existing equipment. Thus, PL/1 may become the main programming language in general use and the single language into which the computing industry places a substantial developmental effort.

In general, the language is based on FORTRAN, is influenced by ALGOL, incorporates the data handling capabilities of COBOL, and recognizes problems associated with multiprogramming and multiprocessing. PL/1 can be thought of as FORTRAN with added capabilities. Some of the more important capabilities that have been added include:

1) File Handling. Files may be described, created, and edited. The structure (item size, identification and format) may be defined so that the item attributes need not be considered in later data translation and numeric computation operations. Items may be stored and retrieved with commands such as GET, PUT, MOVE and SEARCH.

2) Time Sharing. Programs written for PL/1 are assumed to be executed under the control of an executive program. The syntax of the language is such that execution of a program in a time shared environment is possible. Conversely to date, the executive has not been defined and the language contains no instructions to the executive.

3) Data Base Definition. Quantities used in computation (variables) may be defined to be global, local or own. Global quantities have the same value and are located in the same place for all subprograms which use them. Local quantities may be referenced only by the subprograms for which they are defined and during the execution of that subprogram. Own quantities are used to communicate information to subroutines of the subprograms. Global quantities, then, may be thought of as items of a COMPOOL for the program in question.

PL/1 is an advanced language for the general user because it allows him to write programs to be operated in a time shared environment without being required to be aware of memory or storage allocation procedures.

Other significant advantages of PL/1 as they apply to MTACC are:

1) Character manipulation
2) Data directed I/O
3) Real time programming features such as provision of automatic handling of reentrant procedures
4) Operations on bit-strings

## Appendix E

## GLOSSARY

The following is a glossary of terms used throughout this report. The terms are grouped by function only. These definitions were extracted from the IFIP/ICC Vocabulary, June, 1964 edition.

| | |
|---|---|
| OPERATION CODE | A code used to represent the elementary operations of a computer. |
| INSTRUCTION | A general term for a string that specified partially, or completely, an operation or a unit portion of a process. This specification is capable of being used, possibly in conjunction with other data, to cause that operation to take place. In some programming languages the term STATEMENT is used for certain types of instruction. |
| COMPUTER INSTRUCTION | An instruction that specifies a computer operation. <br><br> NOTE <br><br> What is specified by the computer instruction and the way in which it is executed are not under control of the programmer, since such instructions are inherent in the structure of the computer. |
| INSTRUCTION SET | The set of all the different instruction types permitted by a particular programming language. Some programming languages are of such a nature that the concept of an instruction set does not apply. |
| INSTRUCTION CODE | A code used to represent the instructions of a programming language. |
| COMPUTER INSTRUCTION CODE | An instruction code for computer instructions. |

| | |
|---|---|
| COMPUTER INSTRUCTION SET | The instruction set of a computer language. |
| INSTRUCTION FORMAT | The allocation of the characters comprising an instruction between the component parts of an instruction, e.g., the address part, operation part. |
| OPERATION PART (Function Part) | That part of an instruction which specifies the operation to be performed. |
| ADDRESS PART | The part of an instruction which specifies an address partially or completely. An instruction may have several address parts which may specify addresses of various locations. Such instructions are called MULTIPLE ADDRESS INSTRUCTIONS OR MULTI-ADDRESS INSTRUCTIONS. |
| INSTRUCTION WORD | A word, part or all of which is executed by the computer as an instruction. |
| INSTRUCTION ADDRESS | The address of the location where an instruction word is stored. |
| REGISTER | A store or part of a store usually having a capacity of one word, and generally intended for some special purpose or purposes in a computer, which has no address and therefore can be specified only implicitly (or even not at all) by computer instructions. For example, in one-address computers, the accumulator is generally a register and is always implicitly specified by the operation part of computer instructions affecting its content. |
| SHIFT REGISTER | A register adapted to perform shifts; e.g., a delay line register whose circulation time may be increased or decreased so as to shift the content; or a register composed of binary storage cells in which bits are transferred from one cell to the next by the application of a pulse common to all cells. |

| | |
|---|---|
| INSTRUCTION REGISTER | A register in the control unit which stores the current instruction of the program so that it may be interpreted by the control unit. |
| ARITHMETIC REGISTER | A register associated with an arithmetic unit which holds the operands and results of arithmetical and other operations. |
| REGISTER LENGTH | The capacity of a register. |
| TRANSLATE | To transform statements from one language (the SOURCE LANGUAGE) to another (the OBJECT or TARGET LANGUAGE) without significantly changing the meaning. |
| SEARCH (TO)<br>SEEK (TO)(deprecated) | To examine a set of items for any that have desired property. |
| TRANSLATING PROGRAM | A program which translates from one language into another language. In the field of programming the term is commonly used in the more restricted sense of the translation from one programming language to another. |
| COMPILING PROGRAM<br>(Compiler) | A program designed to transform (e. g., translate, assemble and structure) programs expressed in terms of one language (e. g., a procedure-oriented language) into equivalent programs expressed in terms of a computer language or a language of similar form. A compiling program may often include an assembly program. |
| AUTOMATIC PROGRAMMING | The use of an automatic data processing system to perform some stages of the work involved in preparing a program. In particular, to translate a program expressed in a procedure-oriented language into a program expressed in a computer language or into a computer-oriented language. |

| | |
|---|---|
| INTERPRETIVE PROGRAM | A program which deals with the execution of a program by translating each instruction of the source language into a sequence of computer instructions and by allowing these to be executed before translating the next instruction. The major characteristic of an interpretive program is that the translation of an instruction is performed each time the instruction is to be obeyed. |
| ASSEMBLY PROGRAM | A program which assembles parts of a program (sometimes including the acceptance or selection of library programs), making the necessary adjustments to cross-references (such as links) and allocating storage as required. It may also provide translation into computer language. |
| LANGUAGE | A general term for a defined set of symbols and rules or conventions governing the manner and sequence in which the symbols may be combined into a meaningful communication. An unambiguous language, intended for expressing programs, is called a PROGRAMMING LANGUAGE. |
| COMPUTER-INDEPENDENT LANGUAGE | A relative term for a programming language which is not a computer language, but which is, however, intended to be translated, in normal practice, to a variety of computer languages. |
| PROCEDURE-ORIENTED LANGUAGE | A relative term for a computer-independent language especially convenient for expressing a process in terms of procedural or algorithmic steps. Examples are a flow diagram, COBOL, FORTRAN and ALGOL. |
| PROBLEM-ORIENTED LANGUAGE | A language designed for convenience of specifying a class of problems, for example, algebra for specifying mathematical problems. |

NOTE

The concepts computer language, computer-oriented language, computer-independent language, procedure-oriented language, and problem-oriented language are not all mutually exclusive.

| | |
|---|---|
| ASSEMBLY LANGUAGE | A programming language which is closely similar to computer language and is designed to be used in conjunction with an assembly program; in particular, the instructions of an assembly language are generally in a one-to-one correspondence although they may be in a one-to-many correspondence with computer instructions. |
| COMPUTER LANGUAGE | A programming language the instructions of which are computer instructions only. |
| COMPUTER-ORIENTED LANGUAGE (Computer-Dependent Language) | A relative term for a programming language, such as computer language or a closely related language. |
| PRAGMATICS | The study of the extent to which practical use may be made of constructions in a language. Example: A program designed to solve a complex problem requiring the full power of an advanced programming language will be syntactically correct if written in accordance with the rules of the language and will have semantic content to the extent that if embodies a method of forming the required solutions, but its pragmatic value will be lessened if the compiling computer available (or the compiling program itself) impose limitations not inherent in the source language. |
| SYNTAC | In a language, the rules for the formation of permissible constructions (e.g., symbol strings or sentences, valid inferences, etc.) without regard to the meaning. |

| | |
|---|---|
| SEMANTICS | The study of the purposeful meaning jointly assigned to constructions in a language by the users of the language, by other statements in the language or by the context. |
| ADDRESS | In automatic date processing, a numeral or other reference that designates a particular part of a store or some other data source of destination. |
| RELATIVE ADDRESSING | A method of addressing in which the absolute address is obtained by means of a given number to the address part of an instruction. The address part of the instruction is known as the RELATIVE ADDRESS. |
| SYMBOLIC ADDRESSING | The method of addressing using an address (known as a SYMBOLIC ADDRESS) chosen for convenience in programming in which translation of the symbolic address into an absolute address is required before it can be used in the computer. |
| INDIRECT ADDRESSING | A method of addressing in which an address part of an instruction specifies another location which contains the address of the location specified. The address part of such an instruction is known as an INDIRECT ADDRESS. |
| REPETITIVE ADDRESSING | A method of addressing in certain computers that have variable instruction format, in which instructions having zero address instruction format refer again automatically to the location by the last instruction executed. |
| ABSOLUTE ADDRESSING | A method of addressing in which the address part of an instruction is the actual address to be specified. The address part in this case is known as the ABSOLUTE ADDRESS. |

| | |
|---|---|
| ALGORITHM | A set of rules for the solution of a problem in a finite number of steps; e. g. , a full statement of an arithmetical procedure for evaluating sine x to a stated precision. |
| HEURISTIC | An adjective used to describe an exploratory method of tackling a problem, in which the solution is discovered by evaluations of the progress made towards the final result; that is, a process of guided trial and error. Heuristic methods are in contract to algorithmic methods. |
| RECURSIVE DEFINITION OF A FUNCTION | A function that is defined in terms of itself, that is, a function defined by a substitution operation in which the operand includes the function.<br><br>In automatic data processing, a RECURSIVE PROCESS is a method of computing the values of a function, taking advantage of a recursive definition of the function.<br><br>NOTE<br><br>A distinction may be drawn between recursive and iterative processes. In an iterative process each stage of the process is completed before the next begins, while in a recursive process each stage contains all subsequent stages, so that the first stage is not completed until all other stages have been completed. |
| SEARCH CYCLE | That part of a search which is repeated and which normally consists of locating an item and carrying out a comparison. Thus a search in which each item in a randomly ordered set is examined in turn, until the desired item is found, will take on average a number of search cycles equal to half the total number of items in the set. |

| | |
|---|---|
| DICHOTOMISING SEARCH | A search in which an ordered set of items is divided into two parts, one of which is rejected, and the process repeated until the items with the desired property are found. If the number of items in the set is made even at each step of the process and then divided into two equal parts, the search may be known as a BINARY SEARCH. |
| CHAINING SEARCH | A search of an interconnected set for an item whose key matches the SEARCH KEY. The search key is transformed to yield an initial address. If the contents of this address include a key matching the search key, they also include either the item itself, or the location of the item sought; if not, a further address if found in the contents and the process is repeated until either the item is found or the chain terminates. |

## Appendix F

## SOURCE MATERIAL

The following publications were used as source material in preparing this report.

1. "Why Multicomputers?," W.F. Bauer, Datamation, September 1962.

2. "The Burroughs D825," J.P. Anderson, Datamation, April 1964.

3. "PMR Real-Time Data Handling System, System Description and Programming Concept," Informatics, Inc., Unpublished Report to Headquarters, PMR, 1963.

4. "Generalized Multiprocessing and Multiprogramming," A.J. Critchlow, Proceedings, FJCC, 1963.

5. "Computer Design from the Programmers' Viewpoint," W.F. Bauer, Proceedings, EJCC, 1958.

6. "New Concepts in Computing System Design," G.M. Amdahl, Proceedings of the IRE, May 1962.

7. "Analysis of Computing Load Assignment in a Multi-Processor Computer," M. Aoki, G. Estrin and R. Mandell, Proceedings, FJCC, 1963.

8. "Multi-computer Programming for a Large-Scale Real-Time Data Processing System," G.E. Pickering, E.G. Mutschler and G.A. Erickson, Proceedings, SFCC, 1964.

9. "Design for an Associative Computer," P.M. Davies, Proc. Pacific Computer Conference, Pasadena, California, March 1963.

10. "A Cryogenic Data Addressed Memory," V.L. Newhouse and R.E. Fruin, Proc. Spring Joint Computer Conf., May 1962, p.89.

11. "A Magnetic Associative Memory," J.R. Kiseda, H.E. Seelbach, W.C. and M. Teig, IBM J. Res. and Dev., April 1961, pp. 106-121.

12. Content - Addressable Memory System Report, No. 63-25, Nonr 233(52), R.H. Fuller, Department of Engineering, University of California, Los Angeles, California

13. "An Associative Processor," R.E. Ewing and P.M. Davies, Proc., FJCC, November 1964, pp. 147-159.

14. "An Organization of an Associative Cryogenic Computer," R.F. Rosin, Proc. AFIPS SJCC, May 1962, pp. 203-212.

15. "A Hardware-Integrated GPC/Search Memory," R.G. Gall, Proc. FJCC, November 1964, pp. 159-173.

16. "Associative Memory with Ordered Retrieval," R.R. Seeber and A.B. Lindquist, IBM J. Res. and Dev., January, 1962.

17. "Associative Self-Sorting Memory," R.R. Seeber, Jr., Proc. EJCC, December 1960, pp. 179-188.

18. "A Machine for a General Purpose List Processor," R.L. Wigington, Trans. IEEE on Electronic Computers, Vol. EC-12, Dec. 1963, p. 707-714.

19. "IBM System/360 Engineering," P. Fagg, J.L. Brown, J.A. Hipp, D.T. Doody, J.W. Fairclough, and J. Greene, AFIPS Proceedings FJCC, 1964.

20. "Automatic Assignment of Computations in a Variable Structure Computer," G. Estrin and R. Turn, IEEE Transactions Vol. EC-12, No. 6, Dec. 1963.