CPB 230

GE-225
INTRODUCTION TO GECOM

GENERAL ELECTRIC

COMPUTER DEPARTMENT

GE-225
INTRODUCTION TO GECOM

GENERAL ELECTRIC

COMPUTER DEPARTMENT
PHOENIX, ARIZONA

ERRATA SHEET GE-225 INTRODUCTION TO GECOM

OCT 1962

In the interests of increased efficiency and capability, several improvements have been made to the GECOM system since the publication of the GE-225 Introduction to GECOM manual (CPB 230).

Major changes are mentioned briefly below. More detailed descriptions of these and minor changes are available in the two revised publications:

GE-225 GECOM Language Specifications

GE-225 GECOM Operations Manual

ADDITIONAL FEATURES

Compilation

The current configuration of the GECOM system permits program compilation on GE-225 systems having four, five, or six magnetic tape handlers with commensurate reduction in compilation time.

Relocatable Sections

The GECOM system user can now more readily partition a program into Segments and can thereby compile and test each segment separately. Use of this feature requires an appropriate control routine, which can be a modified version of that used for the main program segment. Segments can be compiled so that they can be relocated in memory when all segments are rejoined into a single program.

Common-Storage

The COMMON-STORAGE Section of the Data Division has been fully refined to provide for the description of data to be stored in memory locations that are reserved for shared usage by two or more program segments.

Nested Segments

Provision is made to allow program segments or sections to contain PERFORM sentences which execute other sections.

"N" Controller Compilation

Compilation can be performed using magnetic tape handlers with one to six magnetic tape controllers, as specified by the GECOM user.

Sequence Check

At the user's option, source program card sequence numbers can be checked.

Control Transfers

At the user's option, control transfers based on the type of current record of an input file (determined by automatic Control Key tests) are provided. These transfers are made using statements similar to the following:

1. GO.....DEPENDING ON RECORD OF file-name.
2. If record-name GO.....

SOURCE PROGRAM DECK SEQUENCE

To facilitate many of the above changes and to provide for future improvements and extensions, the organization of the source program deck has been changed slightly. The Data Division must precede the Procedure Division and the END PROGRAM statement (previously at the end of the Data Division) must now be the last statement in the Procedure Division.

Source programs which were previously compiled can be recompiled (if desired) by inserting the Data Division cards, less the END PROGRAM statement, before the Procedure Division and appending a new END PROGRAM statement to the Procedure Division.

EDITED LIST

Minor changes have been made to the format of the Edited List. For example, the interchanging of Data and Procedure Divisions described above is reflected in the Edited List.

Also, the Edited List now provides a count of 1) the GE-225 words that comprise the required subroutines and supplied program segments, 2) the words generated for the main program, and 3) the total of these two groups of words.

FUTURE CAPABILITY

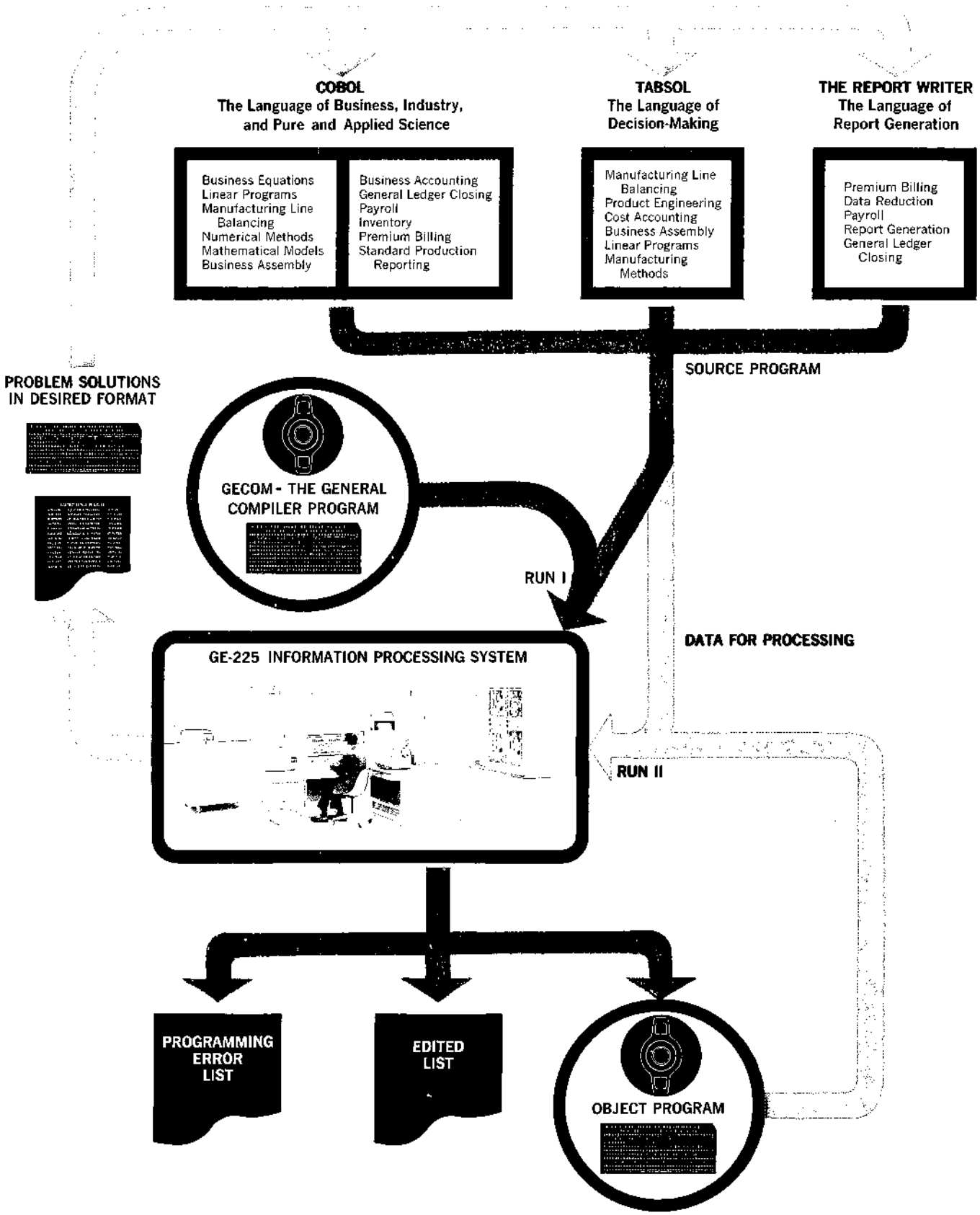
Currently under field test is an extension of the GECOM system which enables the compiler to produce object programs utilizing the 16K memory.

CONTENTS

ILLUSTRATIONS

SOFTWARE MANUALS

GENERAL ELECTRIC reserves the right to make alterations, advances, or modifications to the existing program for reasons of increased efficiency.


COBOL
The Language of Business, Industry, and Pure and Applied Science
Business Equations
Linear Programs
Manufacturing Line Balancing
Numerical Methods
Mathematical Models
Business Assembly
Business Accounting
General Ledger Closing
Payroll
Inventory
Premium Billing
Standard Production Reporting
TABSOL
The Language of Decision-Making
Manufacturing Line Balancing
Product Engineering
Cost Accounting
Business Assembly
Linear Programs
Manufacturing Methods
THE REPORT WRITER
The Language of Report Generation
Premium Billing
Data Reduction
Payroll
Report Generation
General Ledger Closing
SOURCE PROGRAM
PROBLEM SOLUTIONS IN DESIRED FORMAT
GECOM - THE GENERAL COMPILER PROGRAM
RUN I
DATA FOR PROCESSING
GE-225 INFORMATION PROCESSING SYSTEM
RUN II
PROGRAMMING ERROR LIST
EDITED LIST
OBJECT PROGRAM

PREFACE

ABOUT PROGRAMMING

The programming of information processing systems has traditionally been a costly and time-consuming part of automatic data processing. In the past, many applications that otherwise would readily lend themselves to data processing techniques were avoided because of programming costs. Efforts to improve programming techniques have been directed toward producing faster, more economical, and more accurate programs by placing more of the burden on the data processing equipment.

Various combinations of symbolic coding systems (with one-to-one correlation between machine code and symbolic code), macro-instruction coding systems (with a many-to-one correlation between machine code and macro-code), libraries of standardized subroutines, and other innovations were developed to accelerate programming. Despite these improvements, programmers still prepared programs in terms dictated primarily by the computer; programming languages remained essentially machine-oriented languages.

Today, compiler programs provide the programmer with additional leverage. Program coding can be done in a language more suited to the problem instead of in the purely machine-oriented data processor language.

The GE-225 GECOM system, an advanced and effective automatic coding method, provides the next logical step in programming evolution. GECOM is a step toward fulfillment of the much-needed total systems concept--a concept that deems an information processing system to be an integration of application, programming, and information processor or computer.

The GECOM system is further characterized by its applicability to all classes of information processing problems, its ability to grow, and its inherent provisions for use by future General Electric general-purpose computers. GECOM permits coding in the problem languages of business, science, and industry. GECOM can be adapted to future extensions of existing problem languages as the requirement arises, without obsoleting programs prepared to present specifications.

ABOUT THIS MANUAL

This manual is presented as a general information manual about the GE-225 GECOM system and is organized to fill the needs of many people having different levels of familiarity with automatic information processing.

For readers with no previous experience in data processing or computer programming, it is suggested that the entire GE manual be covered. Persons having such previous experience, but who are unfamiliar with the GE-225 Information Processing System, are referred to other General Electric publications, listed below.

Readers already familiar with the fundamentals of programming can begin directly with the section, GECOM Programming Language, with no loss in continuity.

Following the section on GECOM programming language is discussion of the Basic GECOM System. All elements are discussed briefly with the intent of providing overall familiarity with all aspects of GECOM.

The next section treats the two major extensions to GECOM, (TABSOL and the Report Writer), which are first mentioned in the GECOM programming language section, but are more effectively discussed after an understanding of GECOM is achieved.

The reader should not assume that reading this manual will make him a master GECOM programmer. The most effective use of GECOM depends upon training and application. More detailed information concerning the various aspects of the GECOM system can be found in the following General Electric publications:

GECOM	GE-225 Language Specifications GE-225 General Compiler Operations Manual, CD225H1
TABSOL	GE-225 TABSOL Manual, CPB 147 GE-225 Introduction to TABSOL, CPB 147 A
GAP	GE-225 Programming Reference Manual, CPB 126

ACKNOWLEDGEMENT

"This publication is based in part on the COBOL System developed in 1959 by a committee composed of government users and computer manufacturers. The organizations participating in the original development were:

Air Materiel Command, United States Air Force
Bureau of Standards, Department of Commerce
David Taylor Model Basin, Bureau of Ships, U. S. Navy
Electronic Data Processing Division, Minneapolis-Honeywell Regulator Company
Burroughs Corporation
International Business Machine Corporation
Radio Corporation of America
Sylvania Electric Products, Inc.
Univac Division of Sperry Rand Corporation

In addition to the organizations listed above, the following other organizations participated in the work of the Maintenance Group:

Allstate Insurance Company
Bendix Corporation, Computer Division
Control Data Corporation
DuPont Corporation
General Electric Company
General Motors Corporation
Lockheed Aircraft Corporation
National Cash Register Company
Philco Corporation
Standard Oil Company (N. J.)
United States Steel Corporation

This COBOL-61 manual is the result of contributions made by all of the above-mentioned organizations. No warranty, expressed or implied, is made by any contributor or by the committee as to the accuracy and functioning of the programming system and language. Moreover, no responsibility is assumed by any contributor, or by the committee, in connection therewith.

It is reasonable to assume that a number of improvements and additions will be made to COBOL. Every effort will be made to insure that the improvements and corrections will be made in an orderly fashion, with due recognition of existing users' investments in programming. However, this protection can be positively assured only by individual implementors.

Procedures have been established for the maintenance of COBOL. Inquiries concerning the procedures and methods for proposing changes should be directed to the Executive Committee of the Conference on Data Systems Languages.

. . . . . .

Any organization interested in reproducing the COBOL report and initial specifications in whole or in part, using ideas taken from this report or utilizing this report as the basis for an instruction manual or any other purpose is free to do so. However, all such organizations are requested to reproduce this section as part of the introduction to the document. Those using a short passage, as in a book review, are requested to mention "COBOL" in acknowledgment of the source but need not quote the entire section."

INTRODUCTION

WHAT IS GECOM?

The GE-225 GECOM system is an advanced and highly effective method for preparing sets of directions for the GE-225 Information Processing System. As a system, it consists of three elements: Language, Compiler, and Computer. These three terms are further explained below.

THE LANGUAGE

A language is, in general, a means of communication. In the visual form, it usually consists of a set of symbols (such as our alphabet), which can be arranged into meaningful groups (words). Properly arranged aggregates of these groups or words can communicate ideas, action, commands, and questions.

The direction of an automatic information processing system in the performance of a given operation requires communication between man and machine. Just as communication between two men requires a language intelligible to both, communication between man and machine requires a common language. This common language can be machine-oriented (that is, related closely to the basic means by which the computer accepts and presents information, and requiring tedious translation by man of his directions into machine-acceptable form), or the language can be problem-oriented (enabling man to express directions in a form more convenient to the application and placing the burden of the translation on the computer), or it can lie somewhere between these extremes. Machine-oriented and problem-oriented languages are discussed further in the section, "General Programming Concepts".

The GECOM language is a problem-oriented language designed to handle scientific problems as well as general business information processing. The primary basis for the language structure is COBOL, the COmmon Business-Oriented Language for programming digital computers. COBOL is further discussed in the section, "GECOM Programming Language".

In addition to the capabilities derived from COBOL, GECOM language incorporates many of the features of ALGOL, (an ALGOrithmic Language for stating mathematical computations), such as capabilities to evaluate complex equations, Boolean expressions, and mathematical functions. These computations may be performed in either fixed or floating-point arithmetic.

Further versatility is provided by the incorporation of TABSOL and the Report Writer into the language. TABSOL, for TABular Systems-Oriented Language, is a system for expressing decision logic in a simple tabular form. The Report Writer facilitates report preparation and improves documentation. TABSOL and the Report Writer are discussed in the section, "Extensions to GECOM".

GECOM language is not limited to the language capabilities and the extensions mentioned above. General Compiler versatility permits inclusion of GAP, the basic symbolic language (machine-oriented to a degree) of the GE-225 Information Processing System. GAP, for General Assembly Program, is a straightforward symbolic assembly system for the GE-225.

THE GENERAL COMPILER

If communication with the computer is to occur in problem-oriented language, some means must be provided to translate that language within the computer into machine-oriented form. A set of directions for a computer, regardless of the language in which it is prepared, is called a program or, sometimes, a routine. A program, manually prepared, is generally termed a source program. A source program which has been translated into a machine-oriented program is an object program. One means of translating a source program into an object program is to use a specially-prepared program (called a compiler) which, within the computer, operates upon the source program as if it were data and transforms it into an object program.

The General Compiler (from which the GECOM system derives its name) is a unique program specifically designed to reduce sharply the traditionally high programming costs associated with the computer applications. GECOM is a highly versatile and dynamic "program generator"; versatile because it accepts source programs written in a variety of languages; dynamic because both the range of languages and the computer types to which it is applicable can

grow. GECOM represents a major breach in the language barrier between man and machine. Systems and procedures analysts and nonprogrammers can prepare source programs for GECOM translation with only minimal familiarity with the details of machine-language coding. The General Compiler automatically and accurately translates the source program into machine-language instructions.

Compilers and other programming principles are discussed more fully in this section under the heading, "General Programming Concepts".

THE COMPUTER

The GECOM system is initially available for use with the GE-225 Information Processing System, a general-purpose General Electric computer. However, development philosophy was such that GECOM can be implemented on subsequent general-purpose General Electric information processing systems. Thus, as installations outgrow existing equipment, conversion costs for reprogramming are minimized. In addition, existing GE-225 installations who adopt GECOM now, and later expand their system, will have minimal reprogramming requirements. GECOM compiled programs can readily be recompiled to take advantage of expanded machine configurations.

ADVANTAGES OF GECOM

The GECOM system offers many advantages in programming computer applications.

COST REDUCTION

Because of the similarity between the basic GECOM language and ordinary English, costs in many programming areas can be materially reduced:

1. The time and cost of training personnel for program development and maintenance are greatly reduced.

2. Programmers can write more instructions in less time.

3. Programmers can use their time more efficiently by focusing their attention upon the application and results rather than the computer.

4. Easily-read programs reduce desk-checking and debugging time.

5. More machine time is made available for actual processing by reducing machine check-out of programs.

6. Program conversion costs are reduced where installations outgrow present computer equipment or add to existing systems.

FASTER RESULTS

By shortening the time required to prepare programs, GECOM:

1. Reduces the time interval between statement of the problem and the availability of the solution. This provides a faster translation of management requirements into usable information.

2. Makes feasible "one-shot" programs; that is, programs heretofore unwritten because of the high programming costs versus the frequency of use. This would include such applications as investigatory programs.

3. Reduces the overall management information cycle.

IMPROVED COMMUNICATION

Programs written in the modified English of GECOM improve communication in many areas:

1. Programmers can more effectively communicate with other programmers in developing and maintaining programs.

2. Management can communicate more readily with the programming staff.

3. Solutions to problems are delivered in a readily-usable and understandable form, requiring no translation by programmers before being put to use by management.

IMPROVED DOCUMENTATION

Outputs from the program compilation process in a standard and permanent form and standardization of source program preparation facilitate program maintenance and improve programmer versatility.

OTHER ADVANTAGES

In addition to the advantages in the four areas mentioned above, GECOM offers:

1. Convenience - A compact package of programming capabilities called upon by English words and phrases and familiar mathematical symbols.

2. Versatility - One compiler accepts programs written by programming specialists in accounting, mathematic, scientific, or engineering applications.

3. Flexibility - Any or all GECOM features can be selected in appropriate combinations.

4. Adaptability - New advances in automatic coding languages can be incorporated into the "open-ended" GECOM system, as they are proved and implemented by General Electric.

THE INFORMATION PROCESSING SYSTEM

Although the effective use of the GECOM system does not require a detailed knowledge of machine-language programming or data processing systems, some such knowledge is desirable, and perhaps is essential if a valid evaluation of the system is to be made.

Data processing needs have resulted in the development of a great variety of computers. While the physical form and the specific logic flow differ widely, general functions and information flow are similar.

The modern computer or information processor consists of five elements as illustrated in Figure 1: Input, Output, Storage, Arithmetic-Logic, and Control. Communication with the computer is possible only through the input and output elements.

The term, input element, is a functional concept, not the name of a unit of equipment. Only through the input element can data enter the processing system. A system may have one or more of several input media: punched cards, punched paper tape, magnetically-encoded tape, or specially-printed documents. Not all computers have available all input media.

The output element makes it possible for the system to perform a useful function; without an output intelligible to the user, a data processor is useless. Output can take one or more of these forms: punched cards, paper tape, magnetic tape, printing, or any of several special-purpose, machine-controlled forms, such as magnetic-ink encoded (MICR) documents.

Input data must be presented to the system in such a way that the system can manipulate and store it internally. For this reason, data is fed into the system in a form that can be readily converted to the internal electronic language of the system (machine language). Similarly, output data is reconverted to an externally-usable form after processing.

The storage element is functionally subdivided into two general types of storage. One, characterized by limited capacity, high speed, and relatively high cost, is referred to as main storage, memory, core storage, core memory, or simply "core". The latter three terms are popular because tiny magnetic cores are the storage medium in many data processors. The other general type of storage, characterized by high capacity, lower speed, and lower cost, is called auxiliary storage. Auxiliary storage may take almost any form, with punched cards and magnetic tape, discs, and drums being the most common.

The arithmetic-logic element contains the circuits that perform the manipulations of data required by the task or application. It adds, subtracts, multiplies, divides, shifts and rearranges data, and makes decisions, according to the purpose of the program. Capabilities vary widely between different types of computers.

The control element decodes and interprets the stored instructions in proper sequence to achieve the purpose of the program.

In a given compter, it can be difficult to recognize physically the separate storage, control, and arithmetic-logic elements. Functionally, they are separate and distinct elements in all data processing systems and should be so considered. The input and output elements are more readily recognized; more often than not they are packaged as separate units, such as card readers, paper tape readers, document handlers, magnetic tape handlers, card punches, paper tape punches, and printers.

GENERAL PROGRAMMING CONCEPTS

Programming is essentially the framing of a set of directions for a computer. A set of such directions prepared for, and to be communicated to, a computer to guide and control it for a particular processing task is a program.

A subroutine, on the other hand, is a set of directions that is generally incomplete (by itself) in the sense that it usually is only part of a program. Programs frequently contain subroutines for directing the performance of discrete portions of an overall data processing application.

Programs and subroutines, in turn, consist of instructions, which are basic and are the smallest meaningful part of a program. Thus, instructions are the basic tools of the programmer from which he frames the set of directions a computer is to follow.

The phrase "to direct a computer" indicates communication, and communication implies language. In practice, a programmer may use several languages in preparing programs, depending upon the computer. Digital computers are constructed and organized so that they can accept coded representations of letters and numbers, and interpret them as directions to be followed in processing data. Programming languages generally fall into one of three categories, depending on how closely related they are to the computer requirements for accepting information. These three categories are: machine language, symbolic language, and automatic coding language.

MACHINE LANGUAGE PROGRAMMING

Perhaps the most important characteristics of modern information processors is the stored-program concept. In the information processor, instructions

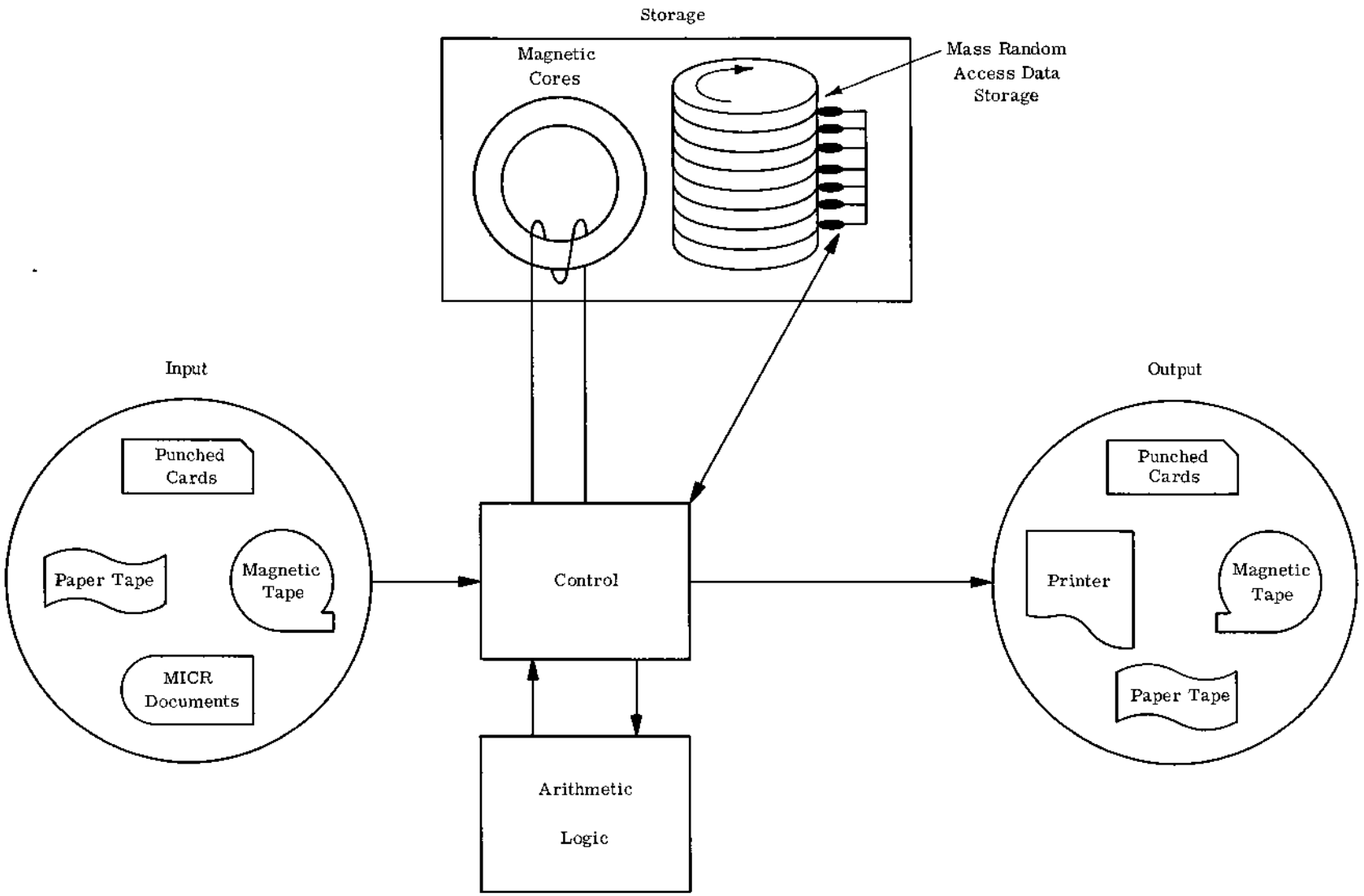


Figure 1. Data Processing Elements

are held in the storage element along with the data to be processed. This not only permits step-by-step data manipulation -- it enables the machine to manipulate its own instructions as if they were data. Thus, it is possible for a program to modify itself (if prepared with this intention) and selectively repeat desired portions.

All information processing systems have a repertoire of permissible instructions; these vary in number and scope from one machine type to another and between manufacturers. For any given system, however, instructions can be grouped by general function:

1. Arithmetic
2. Decision
3. Input/Output
4. Control

Arithmetic instructions, as the name implies, enable the data processor to perform arithmetic such as addition, subtraction, multiplication, and division.

Decision instructions enable the system to compare certain data with some standard (other data, perhaps, or the status of some data processor element) and select alternate courses of action.

Input and output instructions permit the reading in and writing out of data via peripheral input/output units.

Miscellaneous control instructions vary most widely between machines and depend largely upon machine design. In general, simpler machines require more control instructions to accomplish a given function or process than do more complex machines.

Even in the most complex machine, individual instructions are very simple operations and a number of them must be used in the proper order to perform a given function.

For many reasons, most modern information processors are designed to operate internally in some form of the binary (two-digit) number system, or a binary-based system, rather than the conventional decimal (ten-digit) system. Certain computer elements are bi-stable devices (that is: conducting or nonconducting, on or off, open or closed) with the two possible conditions expressed as "0" and "1", corresponding to "off" and "on", respectively. The "0" and "1" represent the two digits of the binary number system and are commonly called bits, for binary digits. By grouping computer elements and assigning values to them according to their position in the group, all numbers may be expressed in binary numbers; for example:

9 = 1001 18 = 10010 523 = 1000001011

wherein the 1-bits, by virtue of their position, have values corresponding to the powers of two (1, 2, 4, 8, 16, 32, 64, 128, 256, 512, etc. from right to left). The 0-bits, of course, as in the decimal system, denote zero value and establish position. Thus, the first 1-bit following the equal sign in the example, 9=1001, has a weight of eight (the third power of two), and the rightmost 1-bit has the weight of one (the zero power of two).

A somewhat similar system permits the representation of alphabetic and special symbols in coded binary form. In fact, the system described so briefly here is only one example of many binary numbering schemes in use and is used primarily to show the concept and illustrate the complexity of programming in a pure machine language. It is rarely necessary to program most modern computers directly in binary or machine language form.

As a final example of machine language programming, a simple routine or program for a hypothetical binary computer is used. Assume that two numbers are in the main storage of the computer at locations arbitrarily called 1000 and 1001. It is desired that the two numbers be added and the result be placed in another storage location, 1002. The binary coding for this program might appear as follows:

(1) 0000000001111101000
(2) 0000100001111101001
(3) 0001100001111101010

The internal computer circuits would interpret such a program thusly:

(1) Load the contents of storage location 1000 into the arithmetic unit.

(2) Add the contents of storage location 1001 to the contents of the arithmetic unit.

(3) Store the new contents of the arithmetic unit in storage location 1002.

Obviously, pure binary programming is slow and tedious, partly because of the difficulty in keeping track of long strings of bits. One innovation that alleviates this difficulty is the use of an intermediate numbering system between the pure binary and the more familiar decimal system.

If the binary numbers in the example above are grouped into three's, as illustrated below, and repetitively assigned the values of the first three

powers of two from right to left, as indicated, then the binary numbers can be represented by a group of seven digits:

	21	421	421	421	421	421	421	← Values of Powers of Two
(1)	00	000	000	001	111	101	000	← Binary Number
(2)	00	001	000	001	111	101	001	← Binary Number
(3)	00	011	000	001	111	101	010	← Binary Number

When evaluated according to the powers of two, these become:

(1)	0	0	0	1	7	5	0
(2)	0	1	0	1	7	5	1
(3)	0	3	0	1	7	5	2

The highest number that can be expressed by any one group of three binary digits is the value 7. To express any value up to 7 requires only eight digits (0 through 7); hence, this numbering system is called the octal system. As can be seen, it is relatively easy to translate from binary to octal and from octal to binary. The translation from octal to decimal and decimal to octal is only slightly more difficult. Because octal provides a more rapid and meaningful way of preparing programs than does binary, on the rare occasion that a programmer needs to work in machine language, he is likely to choose to do so in octal, rather than binary, and to defer translation of his program to binary until it is time to place the program into the computer.

Even with the added facility provided by such aids as the octal numbering system, machine language programming remains slow, tedious, and filled with the possibility of human error. The requirements of knowledge, skill, and patience placed on the programmer are enormous. He must have an intimate knowledge of the internal workings of the computer and its set of instructions in order to take full advantage of its capabilities.

Programming and coding require precision and attention to detail. Every step that the data processor is to perform must be explicitly stated; nothing can be assumed. If two numbers are to be added, the data processor must be told their location, instructed to add them, and told where to place the result. If there is data at that destination, the data processor replaces it with the result of the addition, regardless of the importance of the destroyed data. Thus, it behooves the programmer or coder to keep track of all operations and locations of data in program preparation. The data processor cannot think; it can do only what it is instructed to do.

Program maintenance and revision are difficult in machine language. The addition and deletion of instructions early in the program usually affects the storage locations of subsequent instructions and necessitates modification of addresses. In a program of several thousand instructions, this becomes an almost impossible task and, frequently, it is quicker and less expensive to generate a new program altogether and scrap the old one.

One method of overcoming the shortcomings of machine language programming is through use of symbolic language programming.

SYMBOLIC LANGUAGE PROGRAMMING

It is possible to take advantage of the high speed and the data processing capabilities of a computer to assist in the preparation of programs. One common method is the use of assembly programs. Many of the disadvantages of machine language programming can thus be avoided. One difficulty already mentioned, is the need to include memory, or storage, addresses in instructions. In program modification, if an address is changed, all of the instructions that include the changed address must be found and corrected. Because changes to a program often change many addresses, programs written in machine language are difficult to maintain.

The solution provided by assembly programs is called symbolic addressing. With symbolic addressing, any instruction in a program can be labeled with a symbol that stands for its address, and instructions that refer to it can then be automatically corrected by the assembly program.

An instruction to make a test, for example, can be labeled with the symbol, TEST, and references to this instruction can be by this symbol rather than by its address. When the test instruction is moved to make room for, or delete, instructions before it, any instructions that, perhaps, transfer control of the program to the test instruction will have their transfer addresses changed automatically to the new address of the test instruction.

In addition to symbolic addressing, assembly programs permit other parts of instructions to be symbolized. An abbreviation can be used in place of the machine code for an operation. For example, the abbreviation, SUB, can be used for subtract. This permits more meaningful representation of instructions than does a binary code such as 000010, or an octal code of 02 that might otherwise be used.

The symbolic description of an instruction is converted by the assembly program into an executable machine language instruction. A numeric equivalent is obtained automatically for each symbol occurring in the symbolic address: the memory address is obtained for the symbolic address, the machine code for the operation abbreviation, and so on. The numeric equivalents are then fit together to form a machine language instruction. It is in this area of fitting together that an assembly program assembles.

Descriptions of constants are also accepted by assembly programs. Constants, such as the English word TAX or decimal numbers like 365 are accepted by the assembly program and converted automatically into their machine language equivalents. A legend generally accompanies each description of a constant in the source program to indicate what kind of constant is being described. The legend ALF could be used, for example, to indicate alphabetic constants and DEC for decimal constants.

An assembly program produces the machine language versions of constants and instructions in the object program in such a way that they can be loaded into memory at a later time. Generally, a list is also provided, displaying the symbolic descriptions side-by-side with the output produced in the assembly process for each. The list, called an assembly listing, provides an important documentation of the program. It often contains, also, such aids to program checkout as indications of errors in descriptions and lists of symbolic addresses.

The legends, such as ALF and DEC, that are accepted by the assembly program, but do not stand for actual machine operations, are called pseudo-codes, or pseudo-operations. It is common for an assembly program to provide many of these for the programmer to use. Each extends the ability of the assembly program to prepare or document programs.

The symbolic descriptions of instructions, together with the pseudo-operations that are accepted by an assembly program, constitute what is called an assembly language, or a symbolic language. Although there are numerous exceptions, there is generally one output in machine language for each input in assembly language. For this reason, assembling is often considered to be a one-to-one process.

Symbolic language programming using assembly programs, while considerably simpler and faster than machine language programming, is still highly machine-oriented in that the programmer must have a thorough knowledge of machine-language programming. It is common for source programs written for assembly program processing to result in object programs that are as fast and compact as are equivalent programs prepared directly in machine language. Thus, because symbolic language programs are as efficient as machine language programs, symbolic language programming has almost entirely supplanted the machine language as the basic programming media.

Figure 2 illustrates object program preparation, using an assembly process. First, the programmer prepares the source program in symbolic form, using simple mnemonic codes for the desired machine operations and storage of program constants. Second, the source program is converted to a form suitable for machine entry. The most common representations are hole patterns in punched cards or paper tape or bit patterns on magnetic tape. Usually the programmer prepares his instructions on forms from which a keypunch operator can punch the cards or paper tape for direct entry to the computer or, alternately, for conversion to magnetic tape and the input to the computer.

Next, the assembly program is stored in the computer memory and the source program is input to the computer. The computer, under assembly program control, produces the output -- an object program ready for processing.

At any time after assembly, the object program, now in machine language form, is input to the computer along with data to be processed. The resultant output -- processed data in the form of punched cards, paper or magnetic tape, or printed reports -- is now ready for use external to the computer.

The assembly system available with the GE-225, as previously mentioned, is known as GAP, for General Assembly Program. For further details, refer to the "GE-225 Programming Reference Manual."

AUTOMATIC CODING LANGUAGE PROGRAMMING

As pointed out above, the assembly program permits an already-skilled programmer to prepare programs with a minimum of errors by eliminating many of the details of program "housekeeping." It also provides a more readable version of machine language, thus reducing the need for extensive annotation of machine coding. However, it does not eliminate the need for computer and machine language knowledge.

The compiler program permits the programmer to take another large step away from machine-oriented programming and toward problem-oriented language programming. Compiler programs place even more of the burden of object program preparation on the computer by permitting the programmer to state the desired operations in sentence form or in equation form, depending upon the application and the compiler program.

Compilers have several advantages over assembly programs. The language of the compiler is easier for the programmer to learn and easier for him to use, as it is more closely related to his problem. The programmer using a compiler usually does not need as intimate a knowledge of the inner workings of the computer as does the assembly programmer. Programming is faster; the time required to obtain a finished, working program is greatly reduced because there is less chance for the programmer to make a mistake and because most normal errors are detected by the compiler.

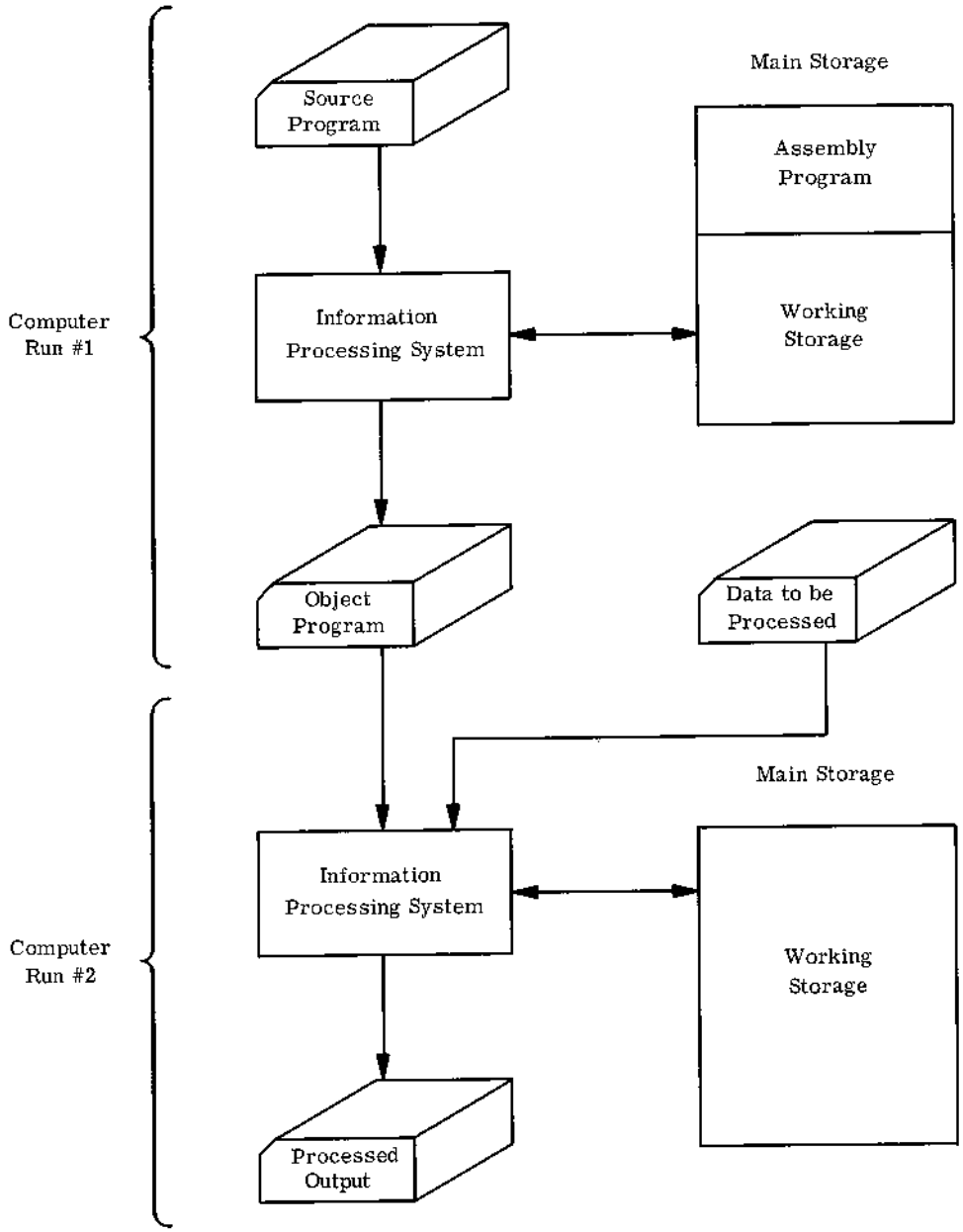


Figure 2. Source Program Processing with Assembly Programs

Advanced compilers are not limited to accepting simply symbolic instructions, but can accept statements approximating ordinary English sentences or mathematical equations. Most of these compilers are highly restrictive in the vocabulary and syntax permissible and in the equipment that can be used. The GECOM system is the first to utilize a General Compiler program to permit both English-language and algebraic programming and, at the same time, to embody provisions for structured decision tables and automatic report writing. Additionally, the General Compiler has built-in provision to expand its language capability to encompass other source languages yet to be constructed.

Many of the advantages of compiler programs, particularly those associated with the General Compiler are pointed out in the section, "Advantages of GECOM". Because the balance of this manual is devoted to describing the GECOM system, it would be redundant to further discuss compilers in general.

However, by virtue of the changing requirements placed upon the programmer who may be engaged in GECOM programming, some consideration should be given to his job title.

The average data processing application involves two broad phases. One phase, defining the problem and determining the general method of solution, is generally called systems analysis. The other phase, involving the actual preparation of the program for computer entry, is variously called coding or programming, although in the strict sense coding is only a subordinate part of programming. In some installations, the two phases are performed by separate individuals; in others, both are performed by one person.

The programmer or systems analyst who is thoroughly trained in GECOM principles can communicate more readily with the computer through the General Compiler and, simultaneously, view the overall application in proper perspective. For this reason, the title, systems programmer, is suggested and used in the balance of this manual to describe the GECOM-trained programmer.

GECOM PROGRAMMING LANGUAGE

GENERAL

All compiler programs accept source programs prepared in specialized language and produce an object program ready for computer processing. Unlike most compilers, GECOM is not restricted to an unduly limited acceptable language. The General Compiler language is actually based on several languages.

The GECOM language evolved primarily from two recent major data processing languages, the business-oriented COBOL and the algorithm-oriented ALGOL. Both languages were developed for solving widely different problems, although from the viewpoint of compiler development they have similar characteristics. These similarities made it possible to provide in one complete and compact package a variety of proven programming techniques. COBOL, which satisfies the needs of the broadest spectrum of data processing applications, provided a basic vocabulary (words and symbols), a basic set of rules of grammer or syntax, and punctuation for clarity. ALGOL, to accommodate the demands of scientific applications, contributes Boolean expressions, floating-point arithmetic, and the ability to express equations concisely.

Many computer applications require neither the extensive file processing facilitated by COBOL, nor the profound mathematics that ALGOL provides, but do involve massive numbers of sequential decisions. To cope effectively with these decisions, General Electric devised structure tables for expressing the relationship of decision parameters. These decision structure tables, and the language in which they are expressed, have been termed TABSOL.

TABSOL has been incorporated into the language accepted by the General Compiler and can be used in combination with the COBOL and ALGOL-like capabilities of GECOM.

In addition to file processing, mathematical applications, and complex decision series, much programming effort is and has been devoted to applications involving report generation. The Report Writer format and language, fully compatible with the General Compiler, gives a fully documented method for preparing reports with minimum programming and debugging effort. The Report Writer is an extension of GECOM and derives much of its advantage from the GECOM system.

Both TABSOL and the Report Writer are discussed in the section, "Extensions to GECOM".

GECOM language is not compartmentalized into the component languages discussed above. In a given source program, it is possible to use COBOL statements containing ALGOL-like algebraic notations; TABSOL decision structure tables can be interspersed with procedure statements; and the Report Writer can be used for report generation. The source program can be prepared using one or all facets of the GECOM language. In addition, if the application so requires, GAP coding sequences can be inserted at will.

COBOL

Because the GECOM language is based primarily on COBOL, some discussion of COBOL and the history of its development is warranted.

In 1959, a meeting was called in the Pentagon by the Department of Defense to consider the desirability and feasibility of establishing a common language for the adaptation of computers to data processing. Representatives from both users and manufacturers were present. The consensus was that the project was definitely both desirable and feasible. As a result, this Conference on Data Systems Languages (CODASYL) established three committees, Short Range, Intermediate Range, and Long Range, to work in four general areas:

Data Description
Procedural Statements
Application Survey
Usage and Experience

In September, 1959, the Short Range Committee submitted a preliminary framework upon which an effective common business language could be built. After acceptance by the Executive Committee of CODASYL, the report was published in April, 1960, by the Government Printing Office as "COBOL-A

Report to the Conference on Data Systems Languages, Including Initial Specifications for a Common Business Oriented Language (COBOL) for Programming Electronic Digital Computers."

A Maintenance Committee of users and manufacturers was created by CODASYL to initiate and review updating changes to COBOL. Using the basic initial specifications published in April, 1960, and recommendations adopted from many sources, in May, 1961, CODASYL approved the publication of the "COBOL-Report to Conference on Data Systems Languages, Including Revised Specifications for a Common Business Oriented Language (COBOL) for Programming Electronic Digital Computers". This report contained specifications for what is commonly called COBOL-61.

COBOL-61 distinguishes between "Required COBOL" and "Elective COBOL". Required COBOL is that group of features and options of the complete COBOL-61 specifications which must be implemented in order to have a COBOL-61 compiler. Elective COBOL, as the name implies, are those features which need not be implemented. However, if they are, implementation should be according to the COBOL-61 specifications.

Present plans of CODASYL include annual revisions to the COBOL specifications in the interests of clarification and expansion with as little effect as possible upon compatibility with earlier COBOL.

Basic COBOL, as defined by CODASYL is a business-oriented language. It is based upon ordinary English and has similar grammer and syntax. Like most problem-oriented languages, programs prepared in COBOL cannot be processed directly by a computer; they must be converted in some way to machine language. CODASYL, in evolving COBOL, limited itself exclusively to language development and left the conversion process in the hands of data processing equipment manufacturers. Manufacturers who have adopted COBOL have prepared compiler programs to make the conversion automatically from source language to machine language.

COBOL is subdivided into four parts:

An Identification Division that identifies the source program and the outputs of a compilation. Optionally, the user can also include the date, author or programmer, and other desired information.

A Procedure Division that specifies the steps the computer is to follow. These steps are expressed in English words, statements, and paragraphs and form the nucleus of the source program. In this division, references can be made, implicitly, to data in other divisions.

A Data Division, wherein files and records that the object program must manipulate or create are described and defined.

An Environment Division, in which the data processing equipment to be used is specified. This division naturally varies widely from computer manufacturer to manufacturer, but generally details memory size, number of magnetic tape units, printers, card readers, and such that are required both to generate and run the object program.

The GECOM language encompasses much of COBOL, as well as certain other languages. Many COBOL capabilities have been adopted and implemented in such a way as to benefit the widest range of users. Programmers familiar with the elements of COBOL will find their training directly applicable to preparing programs for GECOM compilation.

THE BASIC GECOM SYSTEM

GENERAL

For clarity and simplicity, only the Basic GECOM system is described in this section. Brief descriptions of extensions to Basic GECOM are provided in the section, "Extension to GECOM". These extensions, for the most part, expand the capabilities of GECOM to encompass recent language developments.

Implementing a data processing application on a computer involves a broad procedure that has been outlined as follows:

1. Define the problem

2. Determine the procedure to be followed in solving the problem

3. Prepare the computer program, including testing

4. Run the program on the computer with appropriate input data.

If the programmer has at his disposal the automatic coding system of GECOM, the above procedure becomes:

1. Define the problem

2. Determine the procedure to be followed in solving the problem

3. Prepare the source program in problem-oriented language

4. Compile the object program from the source program, using the General Compiler

5. Machine-test (debug) the object program

6. Run the object program on the GE-225 with appropriate input data.

At first glance, automatic coding seemingly complicates the task of data processing. However, as shown in Figure 3, the burden on the programmer is no greater, and often is appreciably less. For example, the step from item 2 to item 3, above, is greatly facilitated by the GECOM-provided ability to express procedural steps in English language statements. Additionally, each statement the programmer writes is several times more powerful than the machine-language or symbolic instructions that he would otherwise use. Also, he is materially assisted in the machine-test or check-out phase, item 5, by the assistance provided by the General Compiler in the form of detailed print-outs of error conditions and of the complete compilation process. The print-outs are as easy to read as the programmer-prepared procedure statements of the source program.

This section is devoted primarily to discussion of item 3, source program preparation, using the GECOM system. Incidental references will be made to the other areas, such as the compilation process, as required.

Assuming that a well-defined data processing problem has been assigned to a systems programmer, he determines the detailed procedures for problem solution and generally prepares a flow chart describing those procedures. Flow charts can be broad or detailed, depending upon the problem and the programmer. Invariably, they are sufficiently detailed to serve as a guide for programming the problem solution. The section, "Application of Basic GECOM," illustrates typical flow charts.

GECOM SYSTEM COMPONENTS

With these preliminaries out of the way, the programmer is ready to prepare the source program. What does the GECOM system provide him to assist in this task?

First, it provides him the necessary language that eliminates tedious machine-language or symbolic coding. Language is discussed in the following section, "GECOM Language Elements".

Second, it provides him with a standard source program organization, which corresponds to the format followed by the compilation output. GECOM source programs are partitioned into four divisions, intended for separate and independent preparation. This facilitates changes; if the procedure must be modified, it can be done with minimal effect upon data parameters; if data changes occur, the data parameters can be changed without affecting the

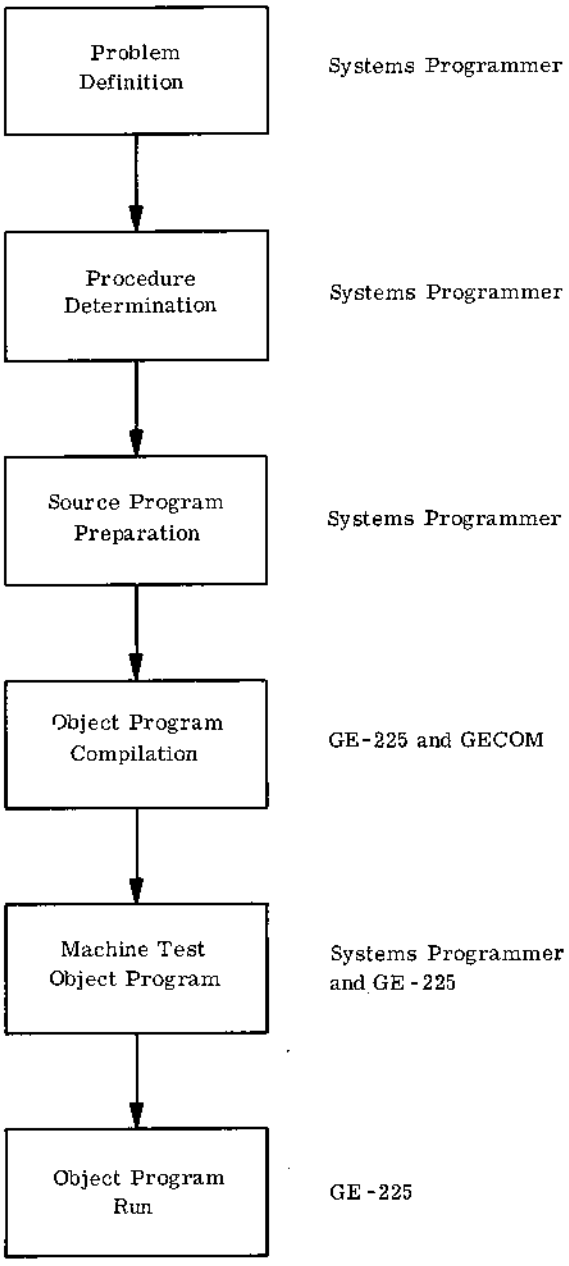


Figure 3. Programming Sequence and Task Assignment

procedure. In addition, standardization of divisions, sections, procedure statements, and other program elements facilitates communication between programmers and permits program debugging in the same language in which the program was written.

The four divisions of a GECOM source program are:

1. The Identification Division
2. The Environment Division
3. The Data Division
4. The Procedure Division

The Identification Division, Figure 4, provides the programmer with the means for labelling and describing the source program in English-language form. In addition to the program name, author (programmer) and date compiled, this division can include other pertinent information, such as next-program-in-sequence, security classification, location, and explanatory comments as needed. During compilation, this data becomes the label for the object program and is automatically reproduced on output listings, such as the Edited List.

Programmer use of the Identification Division is flexible. The only portion required by the General Compiler is the division name and the PROGRAM ID sentence; all other sentences are at the programmer's option.

Preparation of the Identification Division is discussed further in the section, Application of Basic GECOM.

The Environment Division, Figure 5, provides a link between the source program and the data processing equipment. It defines the computer system configuration and its relationship to the source and object program. The General Compiler depends upon the Environment Division to provide information which associates input and output equipment with the data names for each file to be used in processing. The information in the Environment Division is specified by the systems programmer in English language clauses.

In preparing the Environment Division, the programmer enters the information in a predetermined way. This format is sectionalized under four sentence headings as described below:

1. The OBJECT~COMPUTER sentence, the first entry, is used to describe the computer on which the object program is to be run.

2. The I~O~CONTROL (input/output control) sentence, the second entry, specifies nonstandard error and tape label checking procedures. In addition, programming control is facilitated by permitting the specification of program rerun points, memory dump assignments, and identification of multifile magnetic tape reels.

3. The third sentence, FILE CONTROL, identifies input/output files and provides for their assignment to specific input/output units.

4. The COMPUTATION~MODE sentence assigns the internal mode of calculation. Sentence use is optional; it is used only when it is desired that computation occur in the floating-point mode, either programmed or in the optional Auxiliary Arithmetic Unit.

The accompanying example illustrates typical entries describing the environment for a representative program. Entry 10 describes the data processing system for which the object program is intended: a GE-225 system with two memory modules (8192 words of core storage), one card reader, one card

PROGRAM GENERAL REQUISITIONS (8)
PROGRAMMER G. E. CODER

SEQUENCE NUMBER	
1	IDENTIFICATION DIVISION.
10	PROGRAM~ID. REQ~RUN~8.
20	AUTHOR. G. E. CODER.
30	DATE~COMPILED. MAY 10, 1962.
40	INSTALLATION. GE COMP DEPT PHOENIX
50	SECURITY. UNCLASSIFIED.
60	REMARKS. USE DATA FM REQ CARDS.

Figure 4. Identification Division Layout

PROGRAM GENERAL REQUISITIONS (8)

PROGRAMMER G. E. CODER COMPUTER

SEQUENCE NUMBER	
1	ENVIRONMENT DIVISION.
10	OBJECT-COMPUTER. 225, MEMORY SIZE 2 MODULES,
11	1 CR, 1 CP, 5 MT, 1 HSP, 1 PTR, 2 PL,
20	I-O-CONTROL, RERUN ON TAPE 5 ON PLUG 2 EVERY
21	1000 RECORDS OF REQ-FIL,
30	FILE-CONTROL, SELECT REQ-FIL, ASSIGN TO PTR,
31	SELECT REQ-FIL-MT, ASSIGN TO TAPE 1 PL 5,

Figure 5. Environment Division Layout

punch, five magnetic tape handlers, one high-speed printer, one paper tape reader and two controller selector plugs (for connecting the printer and magnetic tape handlers to the GE-225).

Entry 20, the I~O~CONTROL sentence, specifies that restart points are to be established during object program runs each time that 1000 requisition records (REQ~FIL) are read into the computer. On each restart, the contents of core memory are automatically recorded on magnetic tape handler 5, connected to controller selector plug 2. Restarts permit division of object program runs so that, in the event of an error, reprocessing can resume at the last restart point instead of at the beginning, thereby reducing total processing time.

Entry 30, FILE~CONTROL, shows the assignment of the input requisition file (REQ~FIL) to the paper tape reader, and the assignment of the output requisition file to magnetic tape handler 1 on controller selector 5.

Preparation of the Environment Division is discussed further in the section, "Application of Basic GECOM".

The Data Division is used to describe the data to be manipulated or created by the object program. Because data processing frequently involves the repetitive manipulation of similar data, data processing tools are designed with the assumption that information to be processed is logically organized.

For example, to process payroll records, similar data must be available for all employees on that payroll. Each employee has a name, an assignment, a rate of pay, tax exemptions, and other data that make up his pay record. Each piece of data within his record can be called a field. The entire collection of payroll records is usually called a payroll file.

Similarly, a collection of data describing an inventory application can be said to consist of part numbers, item descriptions, unit costs, "used on" codes, and so forth. One part number, its matching description, cost, etc. can comprise an inventory record. Items within the record are fields. The total collection of records would be an inventory file.

Because modern data processing methods are geared to receive data in the form of punched cards or tape and from magnetic fields on tape or specially-coded documents (and thus must be compactly presented), and because the computer itself is an unthinking plodder suited only to repetitive tasks, data must be fed into it in a consistent format. The Data Division of GECOM provides the programmer with a ready means of communicating that format to the computer and, at the same time, assists the programmer in the heretofore error-ridden task of keeping track of specific values.

The GECOM system assumes that data is organized into:

1. Files. A file is a major body of related data stored in external medium, such as punched cards, paper tape, magnetic tape, magnetic disk memory, etc. Files can be small (a few cards) or large (dozens of magnetic tape reels).

2. Records. A logical subdivision of a file consisting of data common to a subject or key and resembling other subdivisions of the same file in size, scope, content, and arrangement.

3. Groups. Within a record, it is sometimes convenient to treat several items or fields (below) as a unit of data.

4. Fields. Specific items within a record (or within a group within a record).

5. Elements. In a few cases, for convenience, fields are further subdivided into "elements." For example, a part numbering system could be so organized that portions of the part number had added significance. For example: 18253702, NPN Transistor; 18 meaning electrical, 2 meaning a component (not a subassembly), 53 meaning tubes and solid-state devices, and 702 to identify the particular item.

The relationship between these various data levels are readily shown:

```
FILE
    RECORD
        GROUP 1
        GROUP 2
            FIELD
            FIELD
                ELEMENT
                ELEMENT
            FIELD
        GROUP 3
        GROUP 4
```

As mentioned earlier, all data to be used or created by the object program must be defined. A typical Data Division for GECOM is shown in Figure 6, giving representative examples of data definitions. The Data Division for a representative problem is presented and explained in the section, "Application of Basic GECOM". The relationship between Data Division and input data is also shown in Figure 6.

The Procedure Division, Figure 7, indicates the steps that the programmer wishes the object program to accomplish. These steps are expressed in English words, symbols, and sentences that have meaning to the General Compiler. Although the steps described in the Procedure Division closely parallel those of the eventual object program, it is misleading to consider the Procedure Division alone to be the source program. The source program is not complete without Data, Environment, and Identification Divisions.

Sentences in the Procedure Division invariably contain verbs to denote the desired action, names (of data, constants, etc.) or operands to show what is to be acted upon, and various modifiers for clarity. Sentences can be grouped into sections to facilitate reference and permit the performance of a series of sentences out of the normal sequence.

Procedure statements or sentences can be simple:

```
ADD 0.5, RATE OF PAY~FILE.
```

This will create coding in the object program to add the constant 0.5 to whatever value (of the RATE from the PAY~FILE) had been read into the computer. Or statements can be highly complex, involving several clauses and modifiers, such as:

```
IF PART~NUMBER OF MSTR~INVNTRY IS
LESS THAN PART~NUMBER OF TRANSAC-
TIONS GO TO WRITE~MASTER, IF EQUAL GO
TO UPDAT~MASTER, IF GREATER GO
TO NEW~RECORD.
```

This statement would result in object program coding to cause the following:

1. The part number of the master inventory record (previously read in) would be compared with the part number of the current transaction record.

2. If the part number of the master inventory record is:

   a. the lesser of the two, program control is transferred to a routine called WRITE~MASTER, which causes the master inventory record to be written out as part of a master file,

   b. equal to the transaction part number, program control is transferred to a routine called UPDAT~MASTER, which modifies the master inventory record in some manner,

   c. the greater of the two, program control transfers to a routine called NEW~RECORD, which causes a new record to be added to the master file.

Procedure Division sentences are performed in the sequence in which they appear, unless that sequence is modified by a "GO" or a "PERFORM" statement as explained in the next section of this chapter, "GECOM Language Elements".

Typical Procedure Division statements are illustrated in Figure 13. Note that sentences can be named (for reference to them by other sentences) or unnamed. Lines 20, 30 and 70 have been named SENT~1, SENT~2, and SENT~3, although more descriptive names can be assigned at the programmer's discretion. More detailed information for preparing a source program Procedure Division is covered in the section, "Application of Basic GECOM".

In addition to LANGUAGE and ORGANIZATION, the third item that the GECOM system provides for the programmer is a set of forms to facilitate source program preparation and documentation. Two basic forms are provided, the General Compiler Data Division Form, number CA-14, and the General Compiler Sentence Form, number CA-13.

Both forms are designed to make it easy to translate the programmer-prepared source program information into a machine-readable form, such as punched cards or paper tape. Each horizontal line of either form provides for up to 80 units of information, corresponding to 80 punched card columns.

GENERAL ELECTRIC

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

GENERAL COMPILER
DATA DIVISION FORM

PROGRAM | DATE

PROGRAMMER | PUTER | PAGE | OF

SEQUENCE NUMBER	TYPE	DATA NAME	ELEMENT POSITION MS	ELEMENT POSITION LS	DATA IMAGE		
1		DATA DIVISION.					
2		FILE SECTION.					
3		INPUT FILES.					
100	FD	REQ~FIL.					
110	R	MSTR~REC.					
120	F	DATE			9(8) B	(1)	
130	E	MO	1	2			
140	E	DA	4	5			
150	E	YR	7	8			
160	F	REQ~NO			9(9) B	(2)	KEYED TO
170	F	CUST~NO			X(4) B	(3)	CARD BELOW
180	F	CUST~ORD~NO			X(6) B	(4)	
190	G	GROUP~1				(10)	
200	F	S~MAN			9(2) B	(5)	
210	F	QTY			9(3) B	(6)	
220	F	STOCK			X(6) B	(7)	
230	F	UNIT~P			9(3)V9(3) B	(8)	
240	F	AMOUNT			9(4)V99 B(24)	(9)	

(10)

(1) (2) (3) (4) (5) (6) (7) (8) (9)

12 18 62 405165870 R902 8A193C 12 144 SN1A09 2108 302688

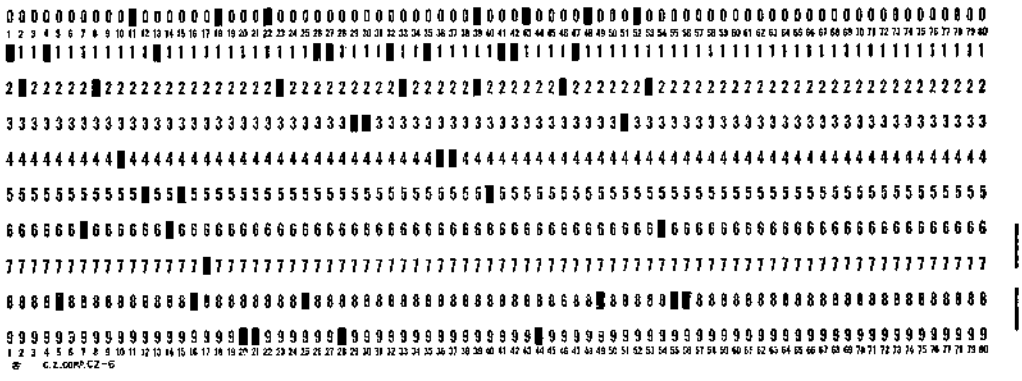


Figure 6. Data Division and Related Input Data

PROGRAM | DATE

PROGRAMMER | COMPUTER | PAGE

SEQUENCE NUMBER

```
1,0   PROCEDURE DIVISION.
2,0   SENT~1.  OPEN INPUT TRANS~FIL MSTR~FIL~IN., OUTPUT MSTR~FIL~OUT HSP~REPT.
3,0   SENT~2.  READ TRANS~FIL.
4,0        READ MSTR~FIL~IN., IF END GO TO FINAL~STOP.
5,0        IF TRANSAC~CODE EQUALS 1 GO TO SHIPMENT, EQUALS 2 GO TO
5,1        RECEIPT, EQUALS 3 GO TO CHANGE, EQUALS 4 GO TO DELETE.
6,0        STOP FIL~MAINT.
7,0   SENT~3.  PERFORM DED~COMP SECTION USING DED OF TRANS~FIL GIVING
               TOTAL~DED.
```

Figure 7. Procedure Division Layout

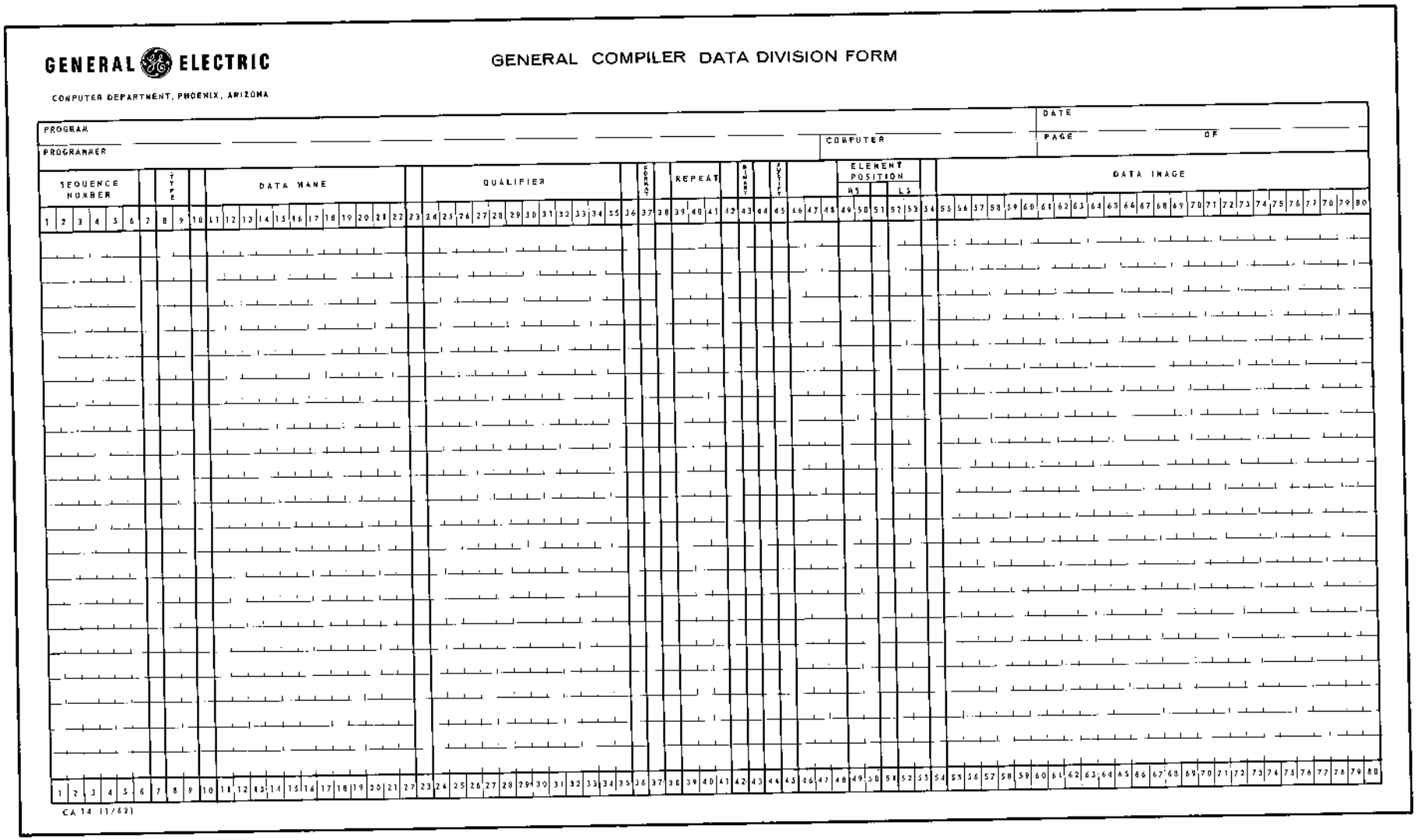


Figure 8. The GECOM Data Division Form

GENERAL ELECTRIC

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

GENERAL COMPILER SENTENCE FORM

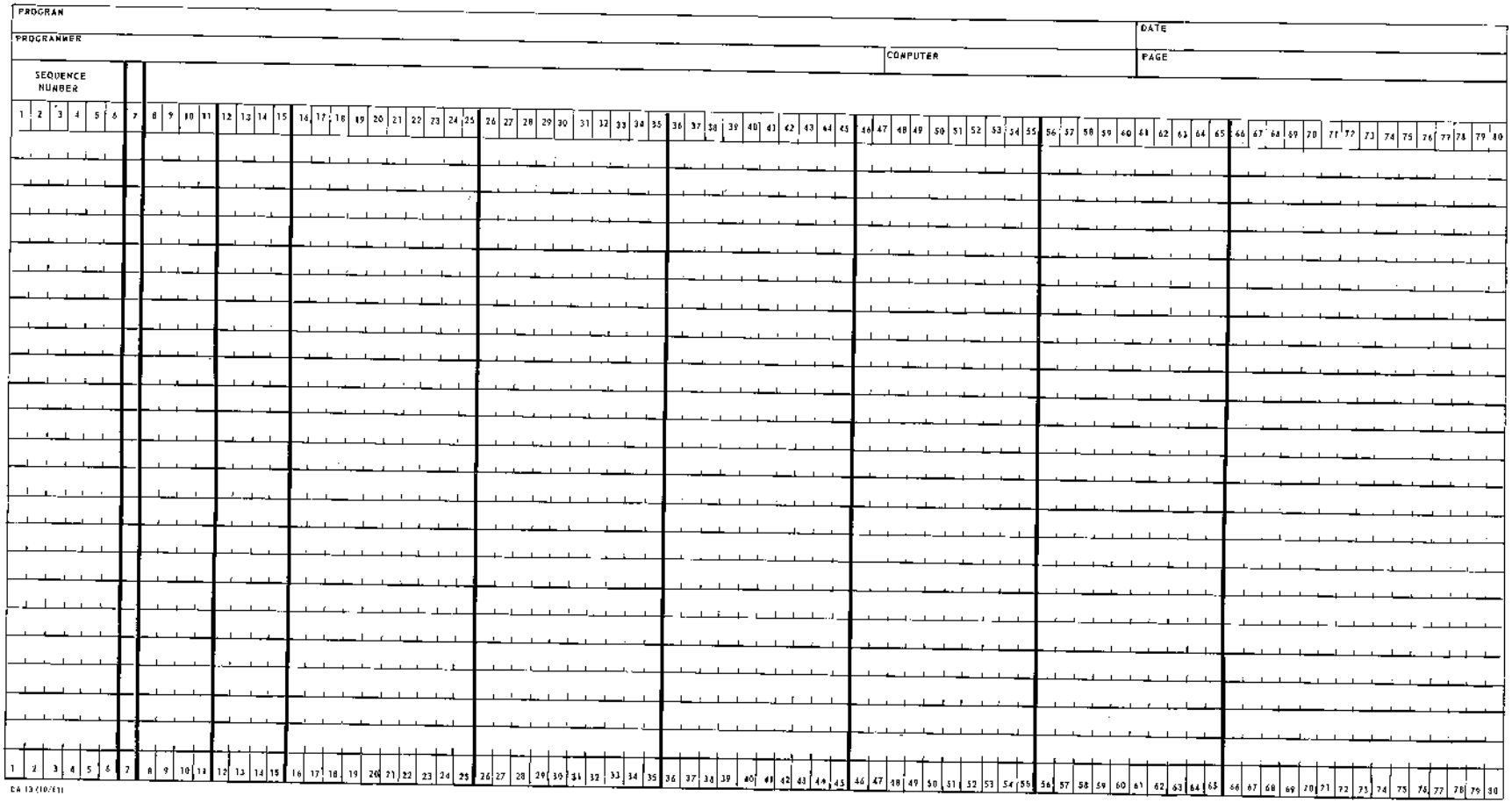


Figure 9. The GECOM Sentence Form

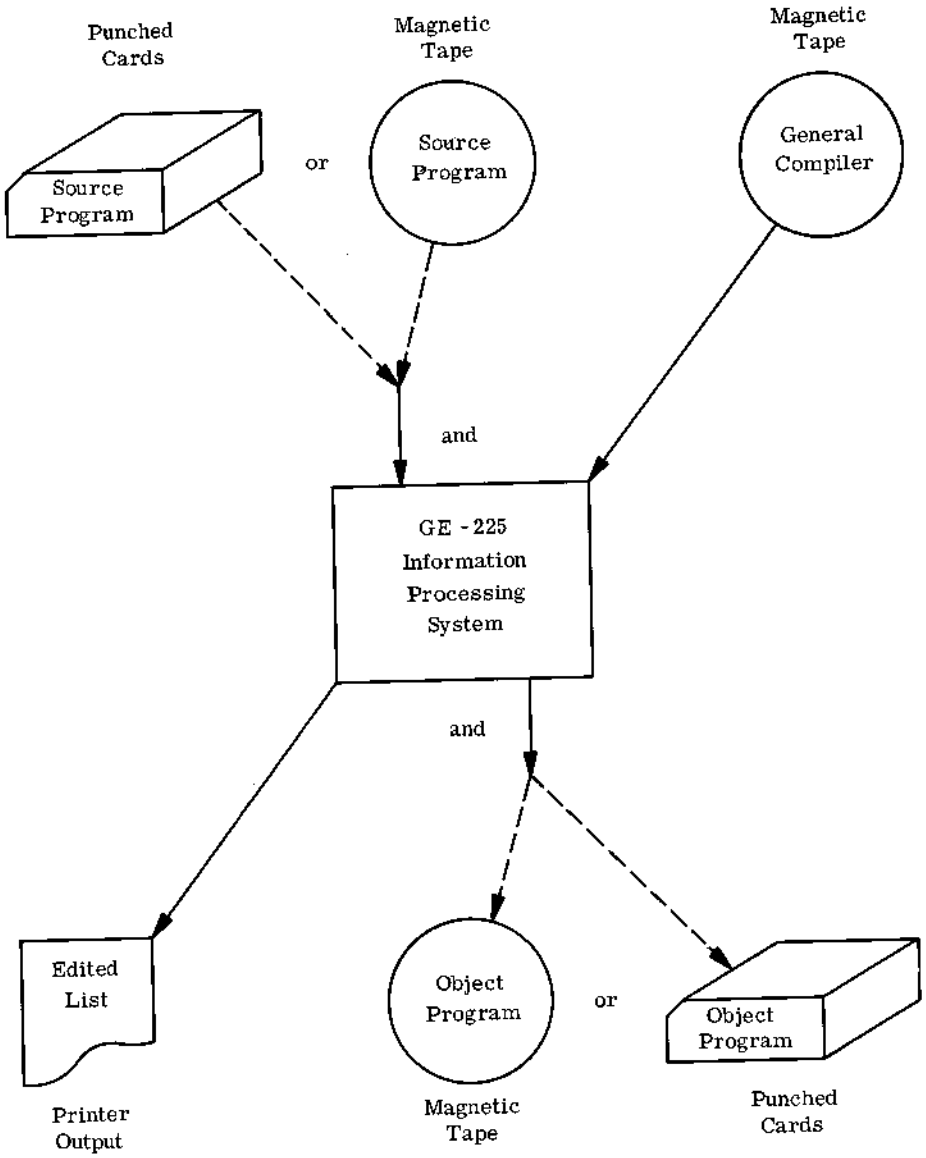


Figure 10. The Compilation Process

The Data Division Form, Figure 8, is used exclusively for describing data to be used in the object program. Headings are provided to guide the proper placement of data. These are discussed in the later section, Data Division Preparation.

The Sentence Form, Figure 9, is used for the preparation of data for the Identification, Environment, and Procedure Divisions. Headings, which would add little, are omitted. Rules for Sentence Form preparation are few and simple.

Where applicable, such rules are discussed in the section, "Application of Basic GECOM," along with the preparation of the four divisions of the source program. The fourth major tool provided by the GECOM system, is the General Compiler itself. Examination shows considerable similarity between the General Compiler program and a complex business data processing object program.

1. The General Compiler operates upon input: the source-language program.

2. Compiler processing consists of repetitive runs of a set of instructions: the General Compiler.

3. It produces an output: the object program.

4. It produces reports: the Edited List and error messages.

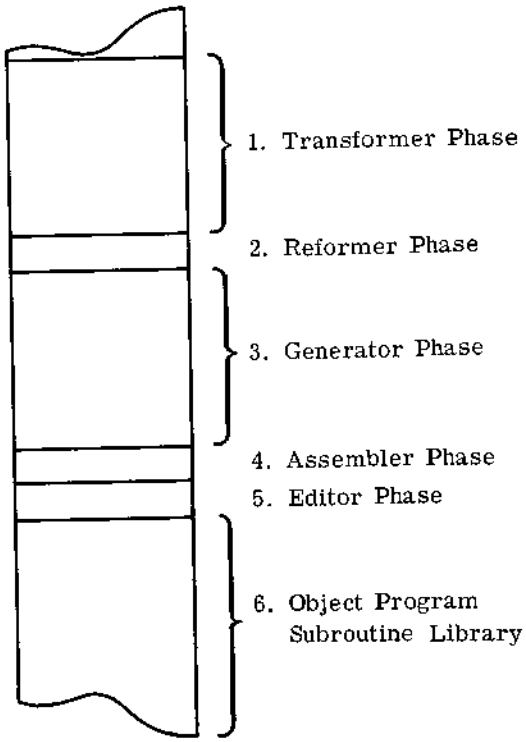


Figure 11. General Compiler Program Organization

Figure 10 illustrates, in broad terms, the relationships between the programmer-produced source programs, the General Compiler, the computer, and the output object program.

Up to this point, the General Compiler has been discussed as if it were a single program, and it can still be considered as such. Conversely, it can also be considered to be a series of sequential programs as illustrated in Figure 11. Note that there are five major groupings: Transformer, Reformer, Assembler, Editor, and Subroutines.

The transformer phase translates the source program into an intermediate internal language suitable for processing, prints out Identification and Environment Divisions as required, groups and organizes Procedure and Data Division material for further processing while checking for validity and consistency, prints error messages, screens out unessential optional words, and initiates the preparation of the object program.

The reformer phase is essentially executive in that it calls forth from the generator library (also a part of the Compiler) those routines required to produce the object program.

The assembler phase translates from the intermediate language, assembles the coding into machine language, and produces the completed object program either in punched cards or on magnetic tape.

The editor phase provides the documentation of the program in the form of the Edited List. This includes a print-out of the entire original source program, a merged list showing the generated symbolic coding and the machine-language coding, and cross-reference tables. Additionally, it lists, from the master list of subroutines below, those required to complete the object program. Examples of the Edited List are included in the section, "Application of Basic GECOM."

The subroutine library is a collection of previously-prepared subroutines common to most object programs that may be required to complete the object program. While these could be produced during compilations, to reduce compilation time and avoid repetitive processing during compiling, the General Compiler shows (on the Edited List) all such subroutines which will be needed when the object program is run. A special program loading routine will place into memory the object program and the

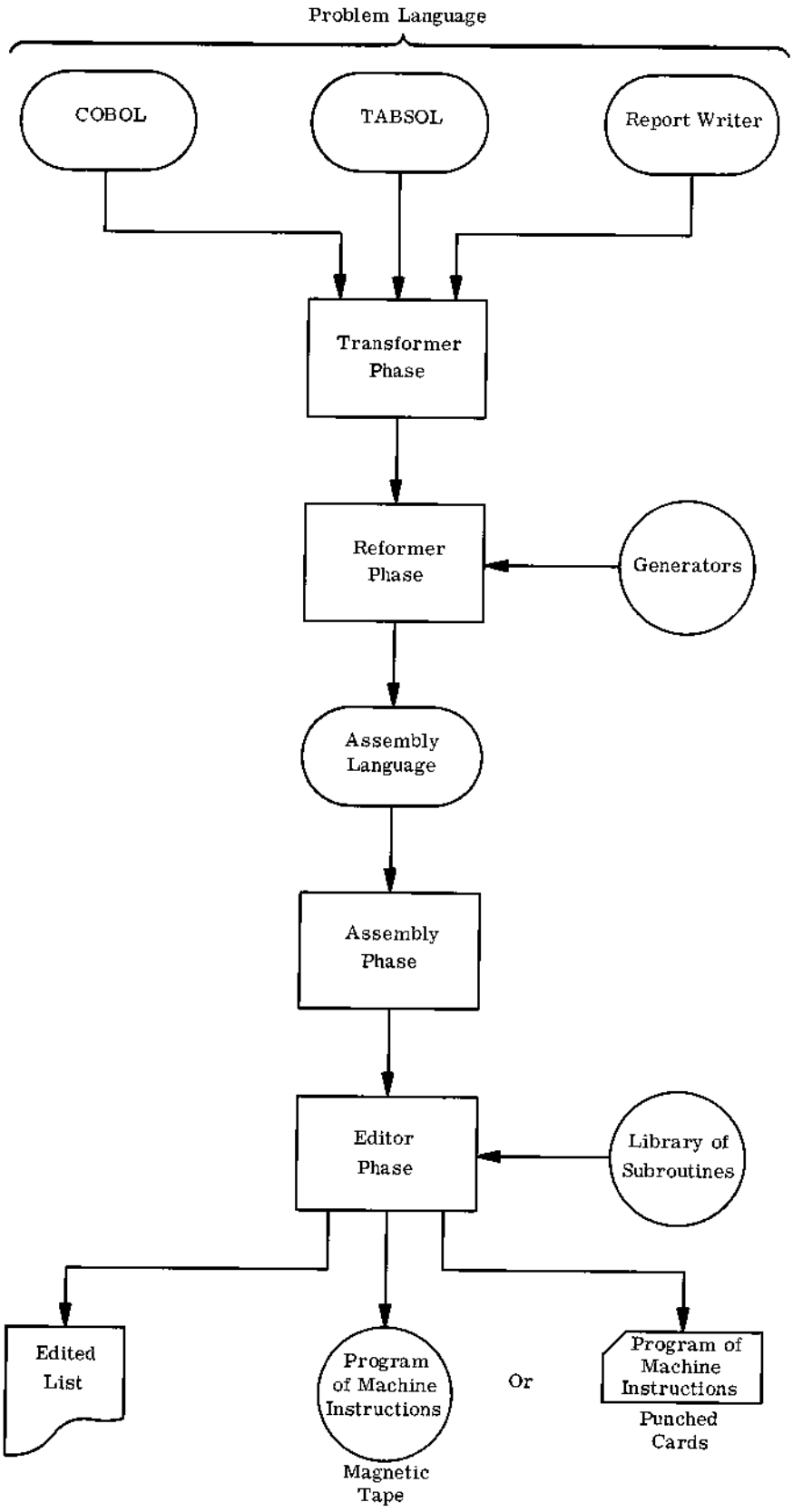


Figure 12. GECOM Inputs and Outputs

required subroutines which the operator has previously extracted from the library of subroutines provided. At the user's option, required subroutines can be appended to the object program automatically or manually during compilations.

GECOM LANGUAGE ELEMENTS

Because the GECOM system was developed with COBOL in mind as the basic programming language, the GECOM language elements most closely resemble those of the COBOL language. Also, because the intent is to provide English-language programming, GECOM elements parallel those of English.

GECOM has a basic vocabulary consisting of words and symbols; it has rules of grammar or syntax; and it has punctuation symbols for clarity. In each case, there is greater simplicity than in English: the vocabulary is small; the rules of grammar are simple, yet precise; the use of punctuation is limited. These are true because the demands placed upon the user are kept simple and unambiguous. The source programming language is required to state facts and give instructions clearly and specifically; it is a language of command, not narration, and thus consists primarily of verbs and nouns. These can be formed into simple and complex sentences usually intelligible without special training, although sentences acceptable to the General Compiler cannot be written without familiarity with the grammar.

Words and symbols are the tools of the GECOM programmer and are composed of individual letters, numbers, and special characters. The basic character set of GECOM and equivalent GE-225 character codes are illustrated in the accompanying table, Figure 13. Special character sets are available for the printer.

Many of the basic characters, in addition to being used in words, have special meanings for GECOM; these will be discussed where appropriate.

Words, in GECOM, are divided into two major groups - names and verbs.

VERBS

As in English, verbs denote action; unlike English, GECOM verbs are never taken in the passive voice, the narrative or declarative sense, or in any tense other than the present tense. Each verb that the programmer uses in the source program (except the verb NOTE) will have some effect in the object program.

Most verbs will be reflected directly in the machine-language coding of the compiled object program; others do not appear in the object program, but do act with the compiler to construct the object program.

Certain words that, in English, are not verbs are considered as such by the General Compiler. The most commonly-used and most useful of these is the word, IF, which is used in expressing conditions, relationships, and comparisons. For example, in the expressions:

IF NOT END OF FILE, GO TO . . . . . .
OR
IF A EQUALS B, GO TO . . . . . .

IF causes a comparison between the actual condition and the stated END OF FILE condition or, in the second example, causes a comparison between A and B. Such near-verbs will be discussed as if they were verbs.

The GECOM verbs and examples of how each might be used are listed in Figure 14.

NAMES

Most words in the GECOM source program will be names. The programmer is preparing a program for handling data, but is not concerned with the actual data itself; he is more concerned with preparing data manipulation procedures, but once they are written they are only of as much importance as the data they manipulate. For these reasons, and to take advantage of the leverage that GECOM provides, the programmer will refer to data and previously written procedures by name whenever possible.

Names can be readily grouped by type and fall within these groups:

1. Data Names
2. Procedure Names
3. Conditional Names
4. Constants

DATA NAMES

Data names represent data to be used in an object program, and are programmer-assigned, not to specific data, but to kinds of data. For example, in a file processing application, data names would be assigned to all input and output files, such as:

MASTER~FILE
TRANSACTIONS
PRINT~FILE
etc.

and, within a file, records would bear data names, such as:

STOCK~RCD
PAY~RCD
INV~RCD~1
etc.

GECOM CHARACTERS	HOLLERITH CARD CODE	GE-225 BCD	MAGNETIC TAPE, BCD	GE-225 PRINTER
A	12-1	010001	110001	A
B	12-2	010010	110010	B
C	12-3	010011	110011	C
D	12-4	010100	110100	D
E	12-5	010101	110101	E
F	12-6	010110	110110	F
G	12-7	010111	110111	G
H	12-8	011000	111000	H
I	12-9	011001	111001	I
J	11-1	100001	100001	J
K	11-2	100010	100010	K
L	11-3	100011	100011	L
M	11-4	100100	100100	M
N	11-5	100101	100101	N
O	11-6	100110	100110	O
P	11-7	100111	100111	P
Q	11-8	101000	101000	Q
R	11-9	101001	101001	R
S	0-2	110010	010010	S
T	0-3	110011	010011	T
U	0-4	110100	010100	U
V	0-5	110101	010101	V
W	0-6	110110	010110	W
X	0-7	110111	010111	X
Y	0-8	111000	011000	Y
Z	0-9	111001	011001	Z
0	0	000000	001010	0
1	1	000001	000001	1
2	2	000010	000010	2
3	3	000011	000011	3
4	4	000100	000100	4
5	5	000101	000101	5
6	6	000110	000110	6
7	7	000111	000111	7
8	8	001000	001000	8
9	9	001001	001001	9
Space, Blank	(Space)	110000	010000	(Space)
. Period, Decimal Point	12-3-8	011011	111011	.
, Comma	0-3-8	111011	011011	,
" Quotation Mark	3-8	001011	001011	#
~ Hyphen	5-8	001101	001101	undrscr
(Left Parenthesis	0-5-8	111101	011101	(
) Right Parenthesis	0-6-8	111110	011110	)
+ Addition, Plus Sign	12	010000	110000	+
- Subtraction, Minus Sign	11	100000	100000	-
* Multiplication	11-4-8	101100	101100	*
/ Division	0-1	110001	010001	/
= Assignment	6-8	001110	001110	=

Figure 13. GECOM Characters and Corresponding Codes

VERB	EXAMPLE
ADD	ADD TOTL~RECVD TO ON~HAND~QTY
ADVANCE	ADVANCE PAY~REGISTER 20 LINES (to slew or skip printer paper)
ALTER	ALTER SENT~25 TO PROCEED TO SENT~33. (to change a previously established sequence of operations.)
=(Assignment)	QTY~ON~HAND = OLD~QTY + NO~RECVD (to assign an evaluated arithmetic expression to a specified field)
CLOSE	CLOSE PAYROL~FILE (to terminate processing of a file)
DIVIDE	DIVIDE NUMBER INTO TOTAL GIVING AVERAGE
ENTER	ENTER GAP AT ROUTINE~3 (to permit insertion of General Assembly Program coding in a GECOM source program.)
EXCHANGE	EXCHANGE OLD~TAX, NEW~TAX (to transpose the contents of two fields)
GO	GO TO SENT~10 (to depart from the normal sequence of operations)
IF	IF LINE~COUNT EQ 58 GO TO ADVANCE~PAGE. (to test a condition and transfer to another operation if condition is satisfied)
MOVE	MOVE TOTAL TO SAVE~AREA (to transfer data to another location)
MULTIPLY	MULTIPLY 0. 18 BY PAY GIVING TAX
NOTE	NOTE THIS SENTENCE IS USED FOR CLARITY. (to permit insertion of explanatory text not intended for compilation)
OPEN	OPEN ALL INPUT FILES (to initiate file processing)
PERFORM	PERFORM FICA~COMP SECTION (to cause execution of a routine in the desired sequence and then return to the sentence following the PERFORM statement.)
READ	READ TIME~CARD RECORD (to make input file records available to the program)
STOP	STOP (to halt processing of the object program permanently or temporarily.)
SUBTRACT	SUBTRACT RECEIPTS OF TRANSAC~FILE FROM ON~ORDER~QTY OF ORDER~FILE GIVING ADJ~ORDER~QTY, IF SIZE ERROR GO TO ZERO~RTN.
VARY	VARY CHK~AMT FROM 1 BY 1 UNTIL CHK~AMT GR 5 (to initiate and control the repeated execution of the sentence it precedes.)
WRITE	WRITE RECORD~1 OF FILE~6 (to permit output of data)

Figure 14. GECOM Verbs

Records are generally further subdivided into items of information called fields. A STOCK~RCD, in an inventory file might consist of these fields:

STOCK~NO
DESCRIPTION
QTY~ON~HND
UNIT~COST
etc.

Fields are also often broken out into elements. The ORDER~DATE field might appear thusly:

MONTH
DAY
YEAR

No advantage is gained in assigning names to lower orders of the data hierarchy than will be referenced in the source program. Hence, data naming is a prerogative of the systems programmer, subject to these provisions:

1. Data names are formed from numerals (0 through 9), letters (A through Z), and hyphens (~).

2. Data names consist of from 1 to 12 characters, at least one of which must be a letter.

3. Data names cannot begin or end with a hyphen (~).

4. Data names cannot contain blanks.

5. Data names must not consist of numerals and the single letter E. (The letter E indicates the exponent in floating-point arithmetic notation.)

6. Data names must not be the same as any of the words comprising the General Compiler vocabulary; see Appendix 1.

In an application involving several files, it is possible that some types of records, groups of records, fields, or elements have the same names. For example, in FILE~1, RECORD~A might contain the fields, ACCT~NO, PAY, and TAX; RECORD~B could have the fields, ACCT~NO, PAY, and TAX. The GECOM system requires that a name be unique either because of spelling or because it can be made unique by association with one or more names above it in the data organization. Thus, to make the TAX field of RECORD A unique, it must be specified in this manner:

TAX OF RECORD~A

If there were no other TAX field in the FILE~1 containing RECORD~A, but there were a TAX field, RECORD~A, in the OUTPUT~FILE, it would be necessary to qualify the TAX field in RECORD~A, FILE~1 thusly:

TAX OF FILE~1

Data names used in this way to make other data names unique are referred to as qualifiers.

PROCEDURE NAMES

For convenience and efficiency, the systems programmer frequently finds it useful to refer to segments or portions of the source program. Such a program segment might be a procedure that he has written for computing FICA tax or a routine for rearranging data within the computer. If such procedures and routines are used repeatedly, the programmer can assign unique names to them and refer to them in the balance of the program as often as required.

Procedure name examples are:

SENTENCE~44
1845
ABC
FICA~CALC

Procedure names are assigned by the systems programmer as needed and are subject to provisions similar to those for data names.

1. Procedure names are formed from numerals (0 through 9), letters (A through Z), and hyphens (~).

2. Procedure names consist of from 1 to 12 characters, all of which may be numerals.

3. Procedure names cannot begin or end with a hyphen (~).

4. Procedure names cannot contain blanks.

5. Procedure names must not consist of numerals and the single letter E.

6. Procedure names must not be the same as any of the words comprising the General Compiler vocabulary; see Appendix I.

CONDITIONAL NAMES

Frequently it is desirable to assign codes to data to conserve space, protect security, or merely for convenient reference. For example, companies often have many pay types represented on the payroll. Some employees are paid according to hourly rates, others are on weekly rates, and still others are paid by the month. Their pay cards might be coded accordingly:

0 = hourly rate
1 = weekly salary
2 = monthly salary

In preparing the source program, the programmer may have difficulty in keeping track of codes that of themselves have no meaning. To provide a reference term, he can assign names to them, thusly:

```
  HOURLY = 0
  WEEKLY = 1
MONTHLY = 2
```

Once names are assigned, they can be used in procedure statements within the source program. Such names as those described above are called conditional names for convenience. In actuality, they are special data names, and are formed subject to the same limitations.

CONSTANTS

Data names are generally assigned by the systems programmer to kinds of data, rather than to specific values, because the actual value of the data named is generally a variable (from record to record, for example) or possibly an unknown to be computed by the object program.

Occasionally (even frequently), the programmer will need to place various kinds of specific data in the program - data which remain the same throughout the program. Such constants are designated as literal constants, numeric constants, and figurative constants.

Literal constants are those the programmer intends to use in the program exactly as written. They may be any combination of up to 30 (or 83, depending upon where used) letters, numbers, and symbols of the GECOM character set. To distinguish them from other names, they must be enclosed in quotation marks:

```
MOVE "FILE~NAME" TO COLUMN~HD.
```

Literals can be used in output fields to generate headings. They cannot be used in arithmetic calculations.

Numeric constants are comprised of the numerals 0 through 9, plus or minus sign, the letter E for floating-point, and a decimal point. They can be used in three forms of arithmetic calculations: fixed-point, integer, and floating-point.

Fixed-point numerics can contain up to 11 digits, excluding plus or minus sign, and a decimal. Typical fixed-point numerics are:

```
+2.308      -853.001
 0.03        9.11
```

Integers must not exceed 5 digits:

```
2308      85300
3         911
```

For floating-point computations, numerics can be written with mantissas of up to nine digits (one of which must be the left of the decimal) and an exponent between +75 and -75. The largest and smallest floating-point numbers that can be represented are, respectively:

```
9.99999999E+75    and    0.00000001E-75
```

If any numeric constant is enclosed in quotation marks, it loses its numeric value and becomes a literal constant.

The constants, 0 through 9 and space (or blank) have been defined within the General Compiler and assigned names. This permits the programmer to use the names within his source program without defining them. These pre-named constants are called figurative constants and are:

```
0 ZERO or ZEROES
  SPACES
1 ONE(S)
2 TWO(S)
3 THREE(S)
4 FOUR(S)
5 FIVE(S)
6 SIX(ES)
7 SEVEN(S)
8 EIGHT(S)
9 NINE(S)
```

Figurative constants may be used in the singular to denote the constant itself or in the plural to imply a string of constants.

EXPRESSIONS

The programmer combines words and symbols into procedure statements to direct computer operations. To facilitate the formulation of such statements showing the relationships and combinations of data names, conditional names, and constants, he has the assistance of arithmetic, relational, and logical expressions.

An arithmetic expression is a sequence of data names, numeric constants, and/or mathematical functions that are combined with symbols which represent arithmetic operations.

Operations and functions available to the programmer and their proper GECOM form are shown in Figure 15. They are listed in priority order, from highest to lowest. All of the listed functions are readily available as part of the GE-225 standard subroutine library and need not be generated during source program compilation or manually by the programmer. Previously-prepared subroutines materially reduce compilation time and programmer effort.

The natural priority of the table can be overridden by parentheses. Parentheses cause the evaluation to be performed from within the innermost set of

OPERATIONS & FUNCTIONS	GECOM SYMBOL
Functions:	
Sine	SIN
Cosine	COS
Arc Tangent	ATAN
Square Root	SQRT
Exponential	EXP
Exponentiation	**
Common Logarithm	LOG
Natural Logarithm	LN
Absolute Variable	ABS
Operations:	
Multiplication	*
Division	/
Addition	+
Subtraction	-

Figure 15. GECOM Arithmetic Operations and Functions

parentheses to the outermost. Up to 50 operations and functions can appear in a single arithmetic expression.

Typical arithmetic expressions might be:

```
        A = X+Y
  RESIDUE = NET~PAY - BAL~DUE
   FACTOR = (A+B*C)*(D-E)
YTD~NET = (YTD~GROSS - RTNS) /22
    ALPHA = EXP (LN(A+B*F))
```

Expressions used to show or imply a comparison of data names, constants, or expressions are called relational expressions. The available relationships that can be expressed are shown in Figure 16.

RELATION	GECOM SYMBOL
Exceeds/Greater than	GR
Less than	LS
Not Greater than	NGR
Not Less than	NLS
Equal (to)	EQ
Not Equal (to)/Unequal (to)	NEQ
Not Positive	NOT POSITIVE
Not Negative	NOT NEGATIVE
Not Zero	NOT ZERO

Figure 16. GECOM Relational Expressions

Typical relational expressions are:

```
IF A+B GR C+D
IF AGE NLS 30
IF GROSS~PAY NEQ 4800
IF (K1+K2)*SIN (A+B) NGR SQRT ORB~VEL
```

Logical expressions provide a means of expressing compound conditions, that is, connecting multiple relational and arithmetic expressions. Connections are made with logical operators, AND, OR, and NOT, and permit compound expressions, such as:

```
NET~PAY AND TAX AND DED LS GROSS~PAY
A-B NLS 50.0 AND R*G GR 272
A+B OR C+D-E AND NOT E+F
```

to be formed. Logical expressions of any size can be used; evaluation is from left to right, with AND taking precedence over OR. Parentheses can be used to establish precedence.

The interpretation placed upon the logical operators is shown in Figure 17, where A and B are variables (or relational or arithmetic expressions) having true and false values.

To illustrate the interpretation of the truth table, consider line 3, which reads: "If A is false and B is true, then Not-A is true and Not-B is false; also A AND B is false (because A is false), and A OR B is true (because B is true).

Stated simply, the operators are interpreted to mean:

1. OR is used inclusively; that is, it is interpreted to mean "either or both".
2. AND means both quantities must be true individually for the entire expression to be true.
3. NOT is exclusive and refers only to the quantity to which it is related. If A is true, then NOT A is false, regardless of the true-false value of other variables; conversely, if A is false, then NOT A is true.

ARRAYS

Tables are used in many clerical operations to provide a convenient means of looking up reference data. Typical of these are the rate books of airlines, railroads, and other transportation services, the logarithm and trigonometry tables of mathematicians, and actuarial tables of insurance companies. Such tables are frequently presented in a ready-reference form such as is shown in Figure 18.

The Left-most column (1, 2, 3, 4, and 5) contains one key for pinpointing data horizontally and the top row (A, B, C, and D) providing the other key. To retrieve the particular fact associated with 3 and C, for example, the clerk merely moves down the left column to 3 and then across to the C column and finds the required data to be C_3.

No.	A	B	Not-A	Not-B	A AND B	A OR B
1.	True	True	False	False	True	True
2.	True	False	False	True	False	True
3.	False	True	True	False	False	True
4.	False	False	True	True	False	False

Figure 17. Logical Expression Truth Table

No.	A	B	C	D
1	A_1	B_1	C_1	D_1
2	A_2	B_2	C_2	D_2
3	A_3	B_3	C_3	D_3
4	A_4	B_4	C_4	D_4
5	A_5	B_5	C_5	D_5

Figure 18. Simple Two-Dimensional Table

Lists and tables of data can be stored within a data processing system for program reference also, permitting the programmer to instruct the program to perform "table look-up" operations. Such tables are stored in series within the system instead of in the grid-like manner illustrated above. The same table in the data processor might appear as a list, shown in Figure 19.

Even though the table data is stored as a long list, the programmer can still readily specify the required table data in essentially the same manner as a clerk would in instructing another clerk how to use the table first shown. The clerk would specify the table name, then the horizontal row and vertical column headings: TABLE 1, row 3, column C. The GECOM programmer does the same thing in a similar shorthand:

```
TABLE~1 (3, 3)
```
meaning TABLE~1, row 3, column 3.

Lists, tables, and matrices can all be represented in GECOM source programs and are referred to generically as arrays. A list is a one-dimensional array; a table, two-dimensional.

A three-dimensional array can be depicted graphically as a series of two-dimensional planes; as shown in Figure 20. Three-dimensional arrays could also be represented in storage as a series of sequential lists (one for each plane) like that described for the example above.

Arrays are assigned identifying names by the programmer. To identify array values, subscripts are used to specify rows, columns, and planes.

```
One-dimensional list = A(I)
Two-dimensional table = A(I,J)
Three-dimensional table = A(I,J,K)
```

Subscripts can be written as arithmetic expressions, if need be, containing other subscripted arrays, and nested to up to ten deep in any one procedure statement.

```
LIST (A+C)
RATE (A-B*C, L(I,J),X)
```

In the second example A-B*C is the i-subscript, L(I,J) is the j-subscript, and X is the k-subscript for a matrix called RATE. Parentheses are always used to enclose subscripts which must immediately follow the array name.

1 A_1 B_1 C_1 D_1 2 A_2 B_2 C_2 D_2 3 A_3 B_3 C_3 D_3 4 A_4 B_4 C_4 D_4 5 A_5 B_5 C_5 D_5

Figure 19. A Two-Dimensional Table in Storage

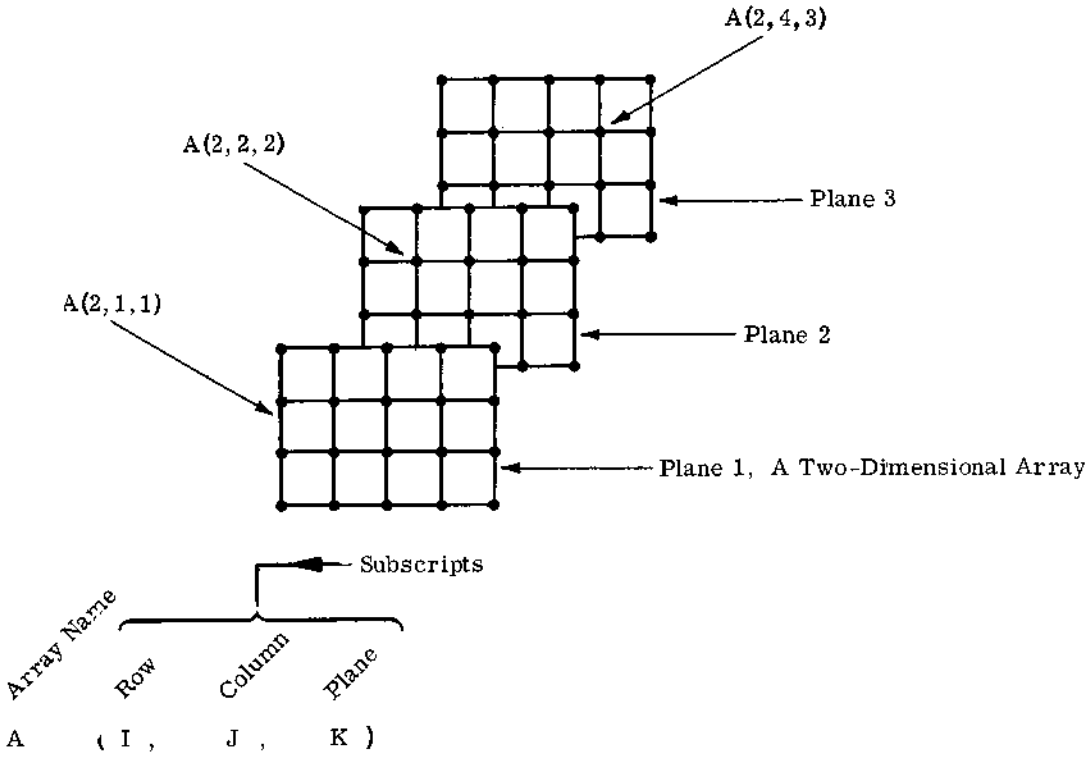


Figure 20. Graphic Representation of a Three-Dimensional Array

EXTENSIONS TO GECOM

GECOM/REPORT WRITER

The GECOM/Report Writer requires the same compiling configuration as Basic GECOM, and is an extension of the basic compiler. Report writing programs can readily be described in the Basic GECOM language, but the Report Writer facilitates report preparation by enabling the user to describe reports concisely on a layout form which can be inserted into the GECOM Data Division. It also provides such features as automatic page and line control, facilitates programming, and provides better documentations of report writing programs.

Report specifications are written within the framework of a GECOM source program, and, in straightforward situations, are contained entirely within the Data and Environment Divisions. A knowledge of file and report formats and which record fields are the file sequence keys is all that is needed beyond a knowledge of GECOM to prepare procedure statements for most business reports. The user need only define the unique features of his job outside of the normal file processing procedure. The Report Writer tailors the basic framework to the programmer's needs and produces an object program for execution. The primary advantages to be gained by this method of description are minimized programming and debugging effort and readily-understandable program documentation.

With proper preparation of the source program, the Report Writer with GECOM will generate an object program which:

1. Prints report headings once at the beginning of the report.
2. Prints report footings once at the end of the report.
3. Maintains page control by line count and skips to a new page as specified.
4. Maintains line spacing on the page.
5. Prints page headings at the top of each report page.
6. Prints page footings at the bottom of each report page.
7. Numbers pages.
8. Issues detail lines according to the presence or absence of control conditions.
9. Accumulates detail field values to one or more levels of total.
10. Counts detail field conditions and detail lines to one or more levels of total.
11. Detects control breaks at one or more levels to control tabulation, issue control totals, and issue control headings.
12. Edits data fields for reporting by zero suppression, character insertion, fixing or floating dollar signs, and fixing or floating arithmetic signs.
13. Assigns and calculates values for report fields.
14. Reads a single file on one or more reels.
15. Reads successive files on multifile reels.
16. Performs normal file opening and closing functions.
17. Creates final totals and terminates reports at end of input.
18. Prepares a report(s) file for deferred printing.

Report descriptions are contained in the Report Section of the GECOM Data Division, under the heading REPORT SECTION, immediately following the File Section. All entries in this section must conform to the format of the Report Description Form, Figure 21, which is used in place of the standard GECOM Data Division form. Not shown are the supporting entries required in the Working Storage Section of the Data Division. Figure 21 illustrates a typical report as laid out in the Report Section of the Data Division, while Figure 22 shows the resulting printed report after processing of the object program containing the report description.

GECOM/TABSOL

The GECOM/TABSOL extension requires the same compiling configuration as Basic GECOM and allows source programs to be described in tabular form. Although the same programs could be described in the basic GECOM procedural sentences, certain benefits are provided by the TABSOL extension.

TABSOL, which stands for Tabular Systems Oriented Language, is basically a structuring technique used to systematically describe the step by step decision logic in the process of solving a problem. The basic advantage of the TABSOL language is that it is easily learned and understood and can be applied to many analytical situations.

GENERAL ELECTRIC
COMPUTER DEPARTMENT

GENERAL COMPILER
REPORT DESCRIPTION FORM

PROGRAM: SAMPLE REPORT, REPORT SECTION
PROGRAMMER | COMPUTER | DATE | PAGE OF

```
SEQUENCE NUMBER | LINE CODE | DATA NAME | CARRIAGE CONTROL
REPORT SECTION.
RD 2    WKLY-PAY                LAYOUT, LINES/PAGE 48.
PH 1                                                          WEEKLY PAYROLL REPORT                          PAGE Z99
PH 2    1                       Z9-99-99
PH 3    1                       ORG     PAY                   JOB          REGULAR      OVERTIME      GROSS
PH 4                            CODE    NUMBER   EMPLOYEE NAME   SEX        CLASS        HOURS         HOURS        EARNINGS
D                               G999.   999999   A A(18)         CCCCCC     XXX          99.9          ZZ.Z         $999.99  X
T       11                      9999.            COUNT OF EMPLOYEES ZZ99                   ZZ999.9       ZZ99.9       $$$,999.99
RD 2                            DEFINITION.
        SELECT                  PAY-DETAIL AND PAY-STATUS EQ NE.
        CONTROL-BREAKS          UNIT, SUBS, SECT, FINAL.
LINE    CONTROL.
T       CONTROL-BREAKS          ALL.
LINE    DEFINITIONS.
PH 1                            1(4) PAGE-COUNT.
PH 2                            MO/D/DA/T/YR.
D                               GRP PAY-NO, 1ST-INITIAL, LAST-NAME, SEX, CLASS, REG-HRS, OVT-HRS, GROSS-EARN, TAG.
T                               GRP LLLL, DETAIL-COUNT, ACC OF REG-HRS, ACC OF OVT-HRS, ACC OF GROSS-EARN.
FIELD DEFINITIONS.
D       GROSS-EARN R            RATE (REG-HRS 1.5*OVT-HRS).
        TAG                     FOR MALE AND GROSS-EARN GR 150.00 OR FEMALE AND GROSS-EARN GR 100.00.
                                114
                                FOR OTHER.
                                SPACE.
T       FOR                     CONTROL-BREAK OF SUBS.
        UNIT                    6.
        FOR                     CONTROL-BREAK OF SECT.
        SUBS                    6.
        FOR                     CONTROL-BREAK OF FINAL.
        SECT                    6.
```

Figure 21. The Report Section of the GECOM Data Division

WEEKLY-PAYROLL REPORT PAGE 28

12-01-61

ORG CODE	PAY NUMBER	EMPLOYEE NAME		SEX	JOB CLASS	REGULAR HOURS	OVERTIME HOURS	GROSS EARNINGS
5484	0671	J JONES		MALE	B01	40.0	10.0	$ 123.44
	0983	A JOHNSON		MALE	A10	37.5		184.01
	1201	B SMITH		FEMALE	C50	40.0	8.0	148.02
	1452	SCHROEDER		MALE	DA2	32.0		84.66
	2352	C BROWN		MALE	D11	40.0	.4	105.19
5484		COUNT OF EMPLOYEES	05			189.5	18.4	645.32
5485	0108	R EDWARDS		MALE	D80	40.0		100.01
	0112	P SMYTHE		FEMALE	B11	35.2		115.55
	1389	A ANDREWS		FEMALE	B01	40.0	8.0	72.06
	1545	R MICHELSON		MALE	A10	40.0	12.0	123.11
	1547	J BERG		MALE	S01	38.2		182.78
	1999	A McMILLAN		FEMALE	C09	40.0	2.2	78.23
	2103	J GWYNN		MALE	B01	40.0	1.8	101.11
5485		COUNT OF EMPLOYEES	07			273.4	24.0	842.85
5480		COUNT OF EMPLOYEES	12			422.9	42.4	1,388.16
5400		COUNT OF EMPLOYEES	33			1302.1	108.0	4,125.29
5501	0133	C STEVENSEN		MALE	E22	40.0		138.06
	0134	L ELLISON		MALE	A09	40.0		149.55
	0222	H MURPHY		FEMALE	C53	40.0		99.99
	2102	J QZER		MALE	B01	40.0		123.02
	2359	A AMBERCROMBIE		MALE	B11	40.0		154.84

Figure 22. Report Writer Sample Report

The tabular technique is not new to industry. Tables have been used for some time as an aid in problem solution. When the manufacturing planner sets up a price table for the planning of coil forming he uses a tabular technique. When the air conditioning design engineer refers to the refrigerant pressure vs. temperature table he is also using the tabular technique to aid in solving the problem. Tables are designed to aid the user in determining specific relational characteristics.

The TABSOL structuring technique involves the use of a table to facilitate the function of specifying decision logic. Computer programming is a perfect example of the job performance that can be improved with the application of this method. The computer programmer receives functional specifications and decision logic from the systems analyst and, in turn, translates this logic into a language that a computer understands. When the programmer speaks to an engineering analyst he must converse in engineering terms. When involved with an accounting analyst a different language is used. The translation of these terms for computer usage generally involves displaying the system logic by means of a flow chart from which the program is written.

TABSOL tables eliminate the need for using flow charts in such a manner and provide improved communication between programmer and analyst.

GECOM/TABSOL which is the union of TABSOL with GECOM, enables the advantages of tabular structured decision logic to be supplemented with all the power of the most up-to-date compiler ever written. This marriage permits the systems analysts to prepare all-inclusive decision tables for direct input to General Electric computers, significantly reducing programming time and effort.

The format of a decision table is given in Figure 23. In concept, a table is an array of blocks divided into four quadrants by a pair of double lines. The vertical double line separates the decisions or "conditions" on the left from the "actions" on the right. The horizontal double line isolates variables from associated operands which will appear in the blocks and rows below. A condition then is a relation between a variable appearing in a primary block and an operand appearing in a corresponding secondary block. For example, AGE may be written in primary block 1 and EQ 26 in secondary block 1. In doing this, a condition is stated: "if age equals 26". An action, on the other hand, is a statement of what is to be done. By writing AGE in a primary action block and 26 in its associated secondary block, it is stated that "the value 26 is to be assigned to age". Note the English interpretation given to the vertical lines. The left-most line may be thought of as representing the word IF. Those lines to the left of the vertical double line may be taken to mean AND; the vertical double line itself, the word THEN. Since actions are sequential entities; the lines separating them may be interpreted as semicolons and the right-most line, which actually terminates the actions, as a period. With this in mind, each secondary row becomes an English sentence. For example, each row now reads:

> "IF condition-1 is satisfied AND condition-2 is satisfied AND.... AND condition-k is satisfied THEN perform action-1; action-2;...; action m."

If any condition within a row is not satisfied, the next row is evaluated and so on until all the rows are depleted. When this happens the table is said to have "no solution". The table is considered "solved" when all the conditions of a row are satisfied and their associated actions performed.

Consider an example to develop an insight into the manner in which TABSOL is used with the General Compiler. The problem is to search a master employee file (recorded on magnetic tape) to determine the number of male employees who fall into the following job categories:

Job Level	Experience (Years)	Title
6	2	Programmer
7	3	Programmer or Analyst
8	More than 3	Analyst
9	More than 4	Analyst or Sr. Analyst
10	More than 4	Sr. Analyst

For each employee found to have these qualifications, his department number, name, title, level and experience are written out on the control console typewriter. At the end of the run the total for each category is also typed on the typewriter.

The core of this problem is the decision that must be made on the information stored in the records of the master file. These decisions are conveniently expressed above in narrative form. With only minor alteration, this form becomes the program statement of the problem. The table and sentences are punched into 80 column cards as they appear in Figure 24. When this is done they may be given directly to the compiler for processing.

As illustrated in the example, General Compiler sentences may be used to support the logic of the table. These sentences accomplish the following:

OPEN (Sequence Number 10) Declares that the MASTER~FILE is input and validates its tape labels.

READ (Sequence Number 15) Delivers the next record from the MASTER~FILE and tests for an end-of-file sentinel. When this sentinel is detected, sequential program execution is interrupted, and control passes to the portion of the program labeled END-RUN.

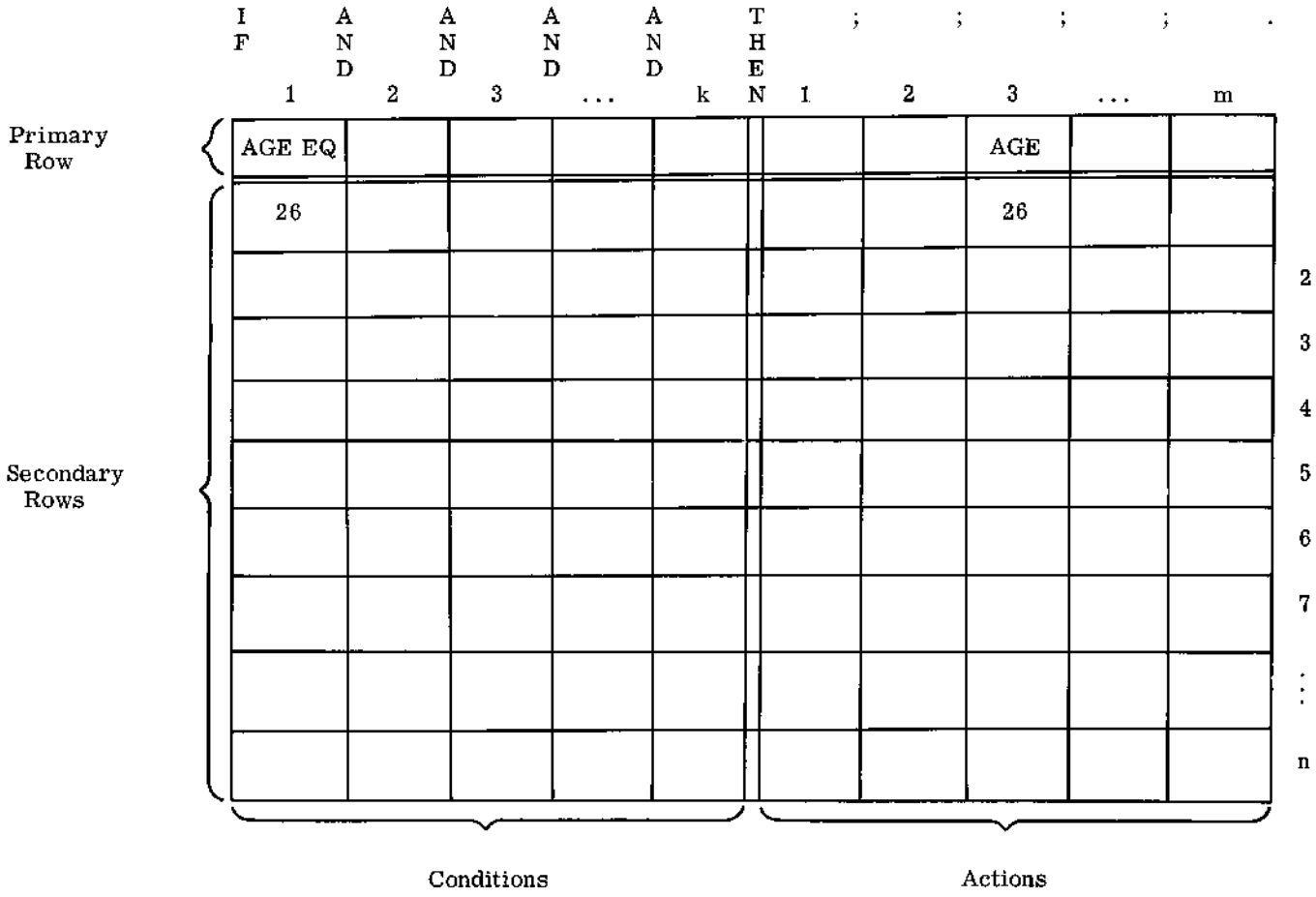


Figure 23. Decision Table Format

IF (Sequence Number 20) Eliminates those data records which contain information about female employees.

EXPERIENCE (Sequence Number 25) Calculates the employee's total experience and assigns the value to the field named EXPERIENCE.

The word TABLE informs the compiler that it must process a decision table; EXAMPLE is a name or label which was given to the table. The size of the table is stated next by giving the number of conditions, actions and rows contained in the table. This information is used only by the compiler and is not executed by the compiled program.

Table execution begins at row 1 (sequence number 40). Using the narrative definition of a table, Row 1 is interpreted as follows: "IF the job LEVEL field equals (EQ) 6 AND the EXPERIENCE field equals (EQ) 2 years AND the employee's title is PROGRAMMER THEN assign the value 1 to the subscript I; GO TO the part of the program having the label TYPE~OUT."

If one of these conditions cannot be satisfied, row 2 is evaluated starting again with the left-most condition. Sequential execution of the rows continues until either all conditions in a given row are satisfied or all rows are exhausted. When the latter situation occurs, the sentence immediately following the table is executed. Proceeding from here, the sentences in the example accomplish the following:

GO (Sequence Number 65) Interrupts sequential program execution and passes control to the part of the program labeled GET~RECORD.

WRITE (Sequence Number 70) Writes the current contents of the DEPARTMENT, NAME, TITLE, LEVEL and EXPERIENCE fields on the computer's typewriter.

TOTAL (I) = TOTAL (I) + 1 (Sequence Number 75) Increments the counter by one.

GO (Sequence Number 80) Passes control to the part of the program labeled GET~RECORD.

CLOSE (Sequence Number 85) Rewinds the MASTER~FILE and performs the file closing conventions.

WRITE (Sequence Number 90) Writes totals for each category on the typewriter.

STOP (Sequence Number 95) Terminates processing and writes the word END RUN on the typewriter.

By General Compiler standards this example represents relatively simple conditions and actions. In formulating these entries, the programmer may take full advantage of the compiler capabilities.

Note the relative ease with which the table can be entered for computer processing. No translation from the problem language via the flow chart to computer language was required. GECOM/TABSOL provides enormous leverage in programming by enabling functional specialists to prepare tables directly for computer entry.

Since its creation, TABSOL has been used by many departments of General Electric to analyze and solve problems in:

Product engineering
Manufacturing methods
Cost accounting
Production control

These areas are not the limitations of possible applications; TABSOL can be applied to all functional operations, such as:

Inventory control
Production scheduling
Shipping and traffic control
Marketing propositions
Marketing requirements
Engineering decisions
Sales programs
Personnel selection

Use of TABSOL is extensive and continues to grow in importance because:

1. Structure tables force a logical step by step analysis of the decision.
2. Structure tables force consideration of all logical alternatives.
3. Structure tables are easily understood and thus form an excellent basis for communication between functional specialists and systems analysts.
4. Structure tables can be written by the functional specialist for direct input to the GE-225 computer, thus reducing computer application costs.
5. Structure tables are easy to maintain, and a system may be easily revised by changing a single value in a single table. In some manual systems inaccuracy is tolerated due to the expense of updated files. This inaccuracy is no longer necessary.
6. The GE-225 electronic computer offers unsurpassed accuracy, ability, and economy in the processing of structure table logic.

The greatest potential of this new language concept lies in the ability to apply it toward the development of a completely integrated business system. The enormous number of daily, routine decisions made by trained and talented personnel are now within the range of mechanization.

COBOL-61/GECOM

While not considered to be an extension to GECOM, the COBOL-61/GECOM Translator is a highly efficient adjunct to the General Compiler. This translator, using the same computer configuration as Basic GECOM, converts source programs prepared according to the latest COBOL Specification, COBOL-61, into language acceptable to the General Compiler.

GENERAL ELECTRIC

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

GENERAL COMPILER SENTENCE FORM

PROGRAM: SAMPLE DECISION TABLE

PROGRAMMER:

COMPUTER:

DATE:

PAGE:

SEQUENCE NUMBER

```
      5  PROCEDURE DIVISION.
     10       OPEN INPUT MASTER~FILE.
     15  GET~RECORD. READ MASTER~FILE RECORD IF END FILE GO TO END~RUN.
     20       IF FEMALE GO TO GET~RECORD.
     25       EXPERIENCE = 61 - YR~EMPLOYED + PREV~EXP.

     30  TABLE EXAMPLE. 3 CONDITIONS 2 ACTIONS 5 ROWS.

     35  LEVEL EQ EXPERIENCE        TITLE                    I  GO TO
     40       6       EQ 2          PROGRAMMER               1  TYPE~OUT
     45       7       EQ 3          PROGRAMMER OR ANALYST    2  "
     50       8       GR 3          ANALYST                  3  "
     55       9       GR 4          ANALYST OR SR ANALYST    4  "
     60       10      GR 4          SR ANALYST               5  "

     65       GO TO GET~RECORD.
     70  TYPE~OUT. WRITE DEPARTMENT NAME TITLE LEVEL EXPERIENCE ON TYPEWRITER.
     75       TOTAL(I) = TOTAL(I) + 1.
     80       GO TO GET~RECORD.
     85  END~RUN. CLOSE MASTER~FILE.
     90       WRITE TOTAL(1) TOTAL(2) TOTAL(3) TOTAL(4) TOTAL(5) ON TYPEWRITER.
     95       STOP "END RUN".
```

Figure 24. Sample TABSOL Table in GECOM

APPLICATION OF BASIC GECOM

GENERAL

To more closely relate the use of the GECOM system to actual applications, the following pages carry a sample problem through the programming process. Although not all of the capabilities of Basic GECOM are exercised, enough material is presented to provide perspective and insight into the scope of GECOM.

First, the problem is presented and the objective is defined.

Second, the procedure to be followed is outlined, the required inputs and desired outputs are identified, and a flow chart is prepared.

Third, the source program is produced. Each of the four divisions of the GECOM source program are illustrated and discussed where appropriate. The compilations and debugging of the object program, performed on the GE-225, are not covered in detail. Procedures for compilation are fully discussed in the GE-225 GECOM Operations Manual, CD 225H1.

Finally, the outputs of the compilation process, the Edited List and the object program, are presented and discussed.

DEFINING THE PROBLEM

The sample problem selected involves a typical manufacturing plant that uses job ticket records for each employee to produce time and job accounting data. Assuming that the individual Job Ticket Records follow the format illustrated in Figure 25, the problem is to prepare a program that will produce two outputs:

1. A punched card summary record for each department, showing the:

   Department Number
   Number of Men
   Accumulated Regular Hours
   Accuumulated Overtime Hours
   Total Hours

2. A printed report providing, by department and man number, this information for each man:

   Department Number
   Man Number
   Name
   Job
   Regular Hours
   Overtime Hours

Figure 26 shows a representative punched card summary record, while Figure 27 shows the desired printed report.

In an actual application, it is quite possible that the input data (the Job Ticket Record) and the desired outputs (the Job Ticket Summary and the Department Man Hour Report) would not already be defined. The problem might be as informally stated as, "we need to know what our people are doing and how long it takes to do it."

In these circumstances, the problem would also entail determining what input data is needed, how to collect it, and how to record it for computer input. It would also be necessary to determine (more precisely than the quoted problem states) what output is desired and what form and organization it should follow.

Here, these preliminary decisions have been made. It remains for the programmer to document the process to be performed by the data processor, detail the procedure the program must follow (via a flow chart), and prepare the source program.

PLOTTING THE SOLUTION

In the sample problem, documenting the process involves little more than translating the problem statement into a diagram. The input is already defined; the purpose of the program has been stated; and the desired outputs have been described. Graphically the process chart appears as shown in Figure 28.

A more realistic application might involve several inputs and outputs via several media. Additionally, multiple "runs" or processes by the data processor

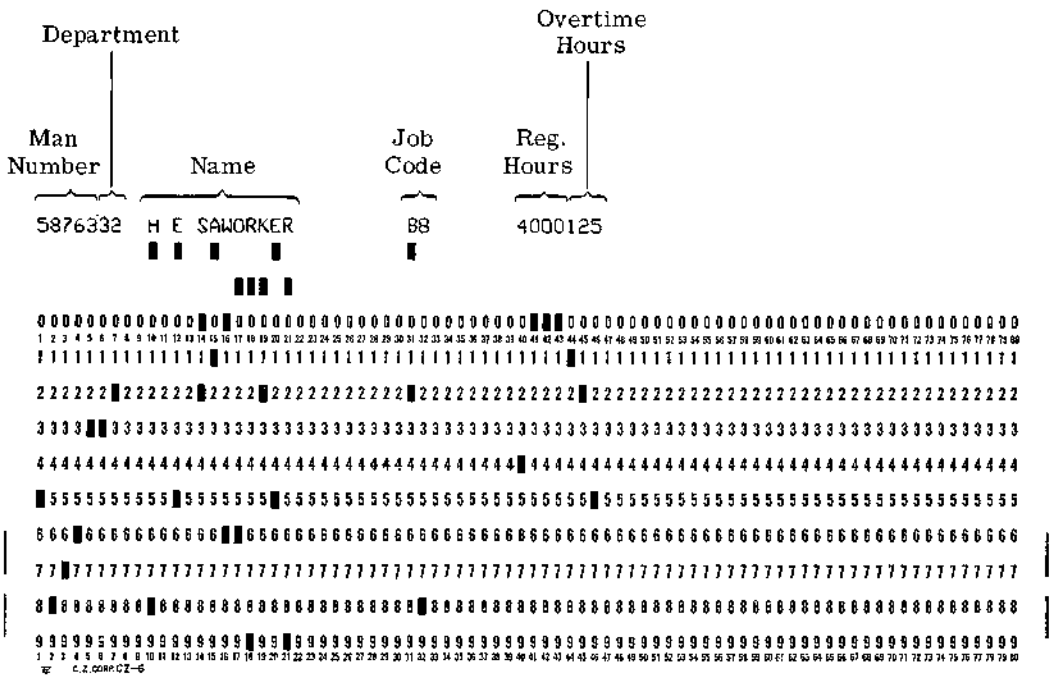


Figure 25. Job Ticket Record Sample

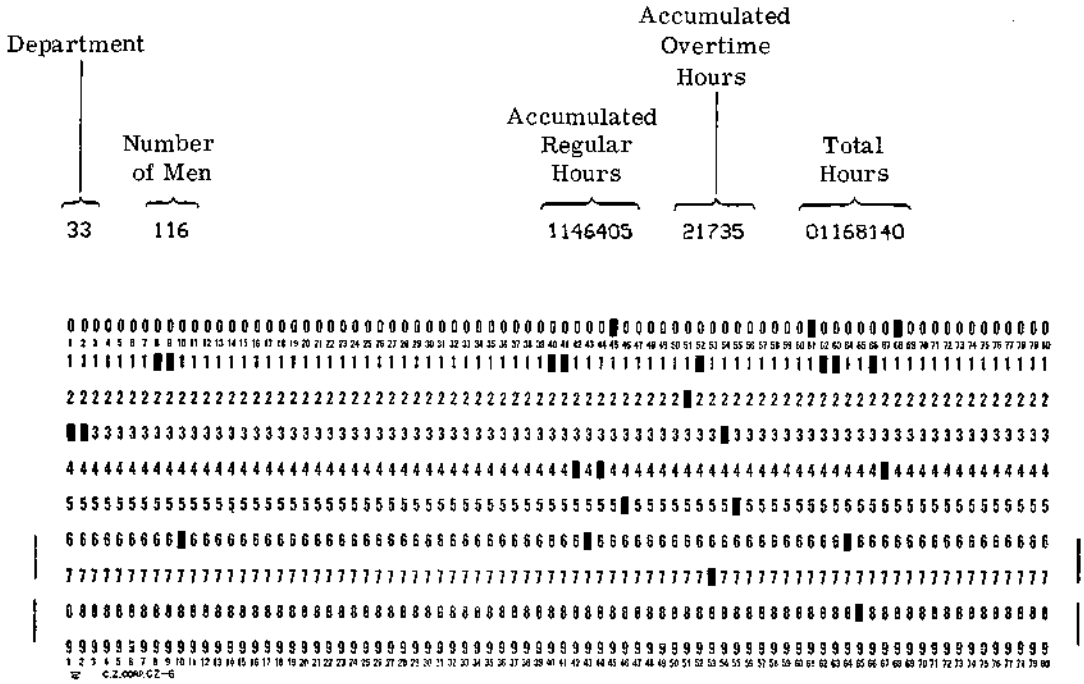


Figure 26. Job Ticket Summary Sample

DEPARTMENT MAN HOUR REPORT PAGE 1

DEPT	MAN NUMBER	NAME	JOB	REG-HRS	OT-HRS
20	10076	FIELY, CR	75	40.0	4.2
	18270	JOHNSON, HA	82	40.0	6.4
	28883	RANGEL, MM	17	40.0	8.6
	30106	STRONG, AB	24	40.0	8.8
	35596	HAYS, ER	33	40.0	2.0

Figure 27. Department Man Hour Report

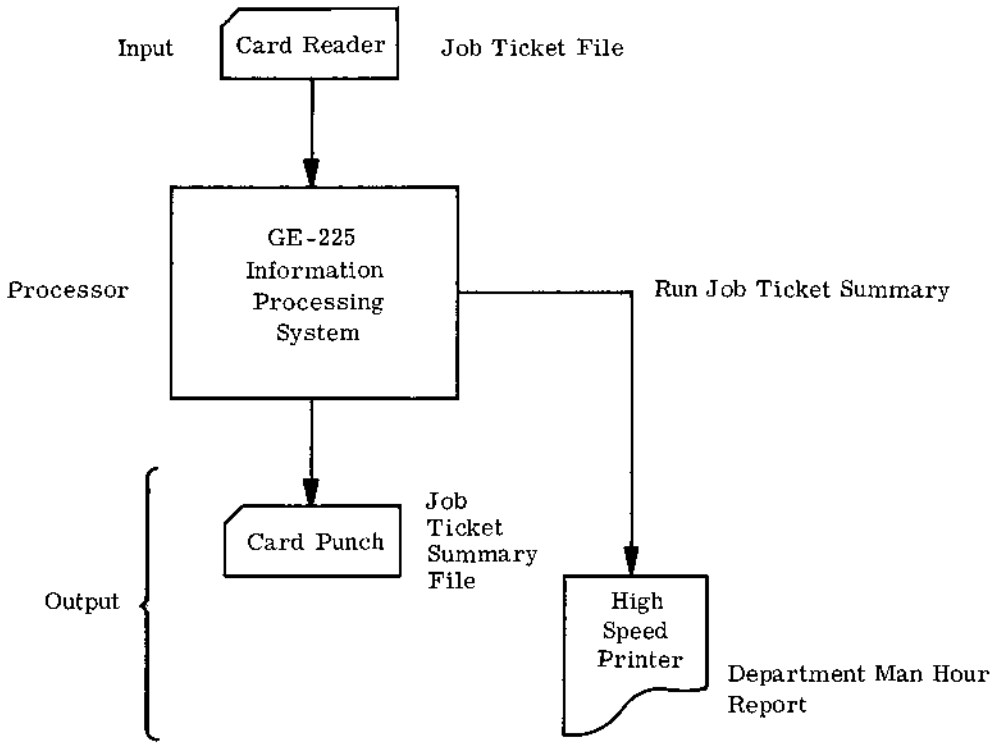


Figure 28. Process Chart for Job Ticket Summary

are not uncommon. To illustrate GECOM effectively, the sample problem is sufficiently complex.

The programmer's next step is to determine the most effective and efficient procedure for accomplishing the goals desired. A flow chart shows the results of his efforts, at the same time providing a tool for planning.

A flow chart, depending upon its level of preparation:

1. Indicates in proper sequence all steps to be followed by the data processor to produce the desired outputs.

2. Provides a graphic representation of the completed project analysis.

3. Anticipates every possible contingency and shows the alternative procedures.

4. Indicates logical decisions, arithmetic computations, data transfers, and other operations essential to problem solution.

5. Provides the programmer with a (more or less) detailed guide to program preparation.

Figures 29 through 31 show one programmer's method for flow charting the Job Ticket Summary problem. While correct, it is not necessarily the best or the only correct procedure to follow. Just as there are many ways to travel from point A to point B in a given city, there are many methods for flow charting and programming a given problem.

A detailed explanation of flow charting or of this particular application is not essential to understanding GECOM. However, a close comparison between the flow charts in Figures 29 through 31 and the Procedure Division in Figure 34 will show how closely the GECOM procedure statements follow the procedure that the flow charts outline. Additionally, flow charts and Procedure Division are complementary insofar as understanding the application are concerned. GECOM sentences are intelligible in themselves; but the flow charts provide overall perspective.

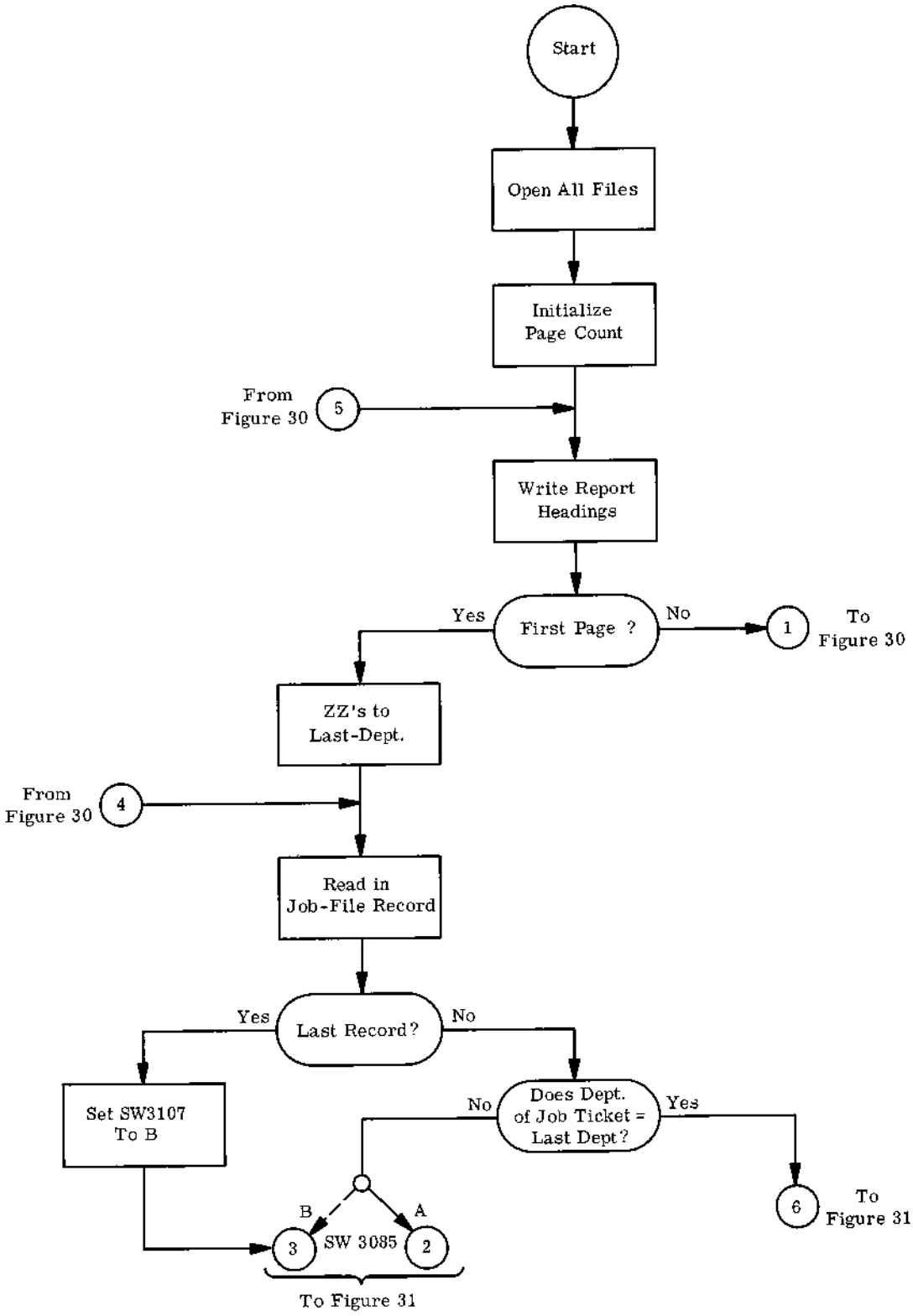


Figure 29. Job Ticket Summary Flow Chart

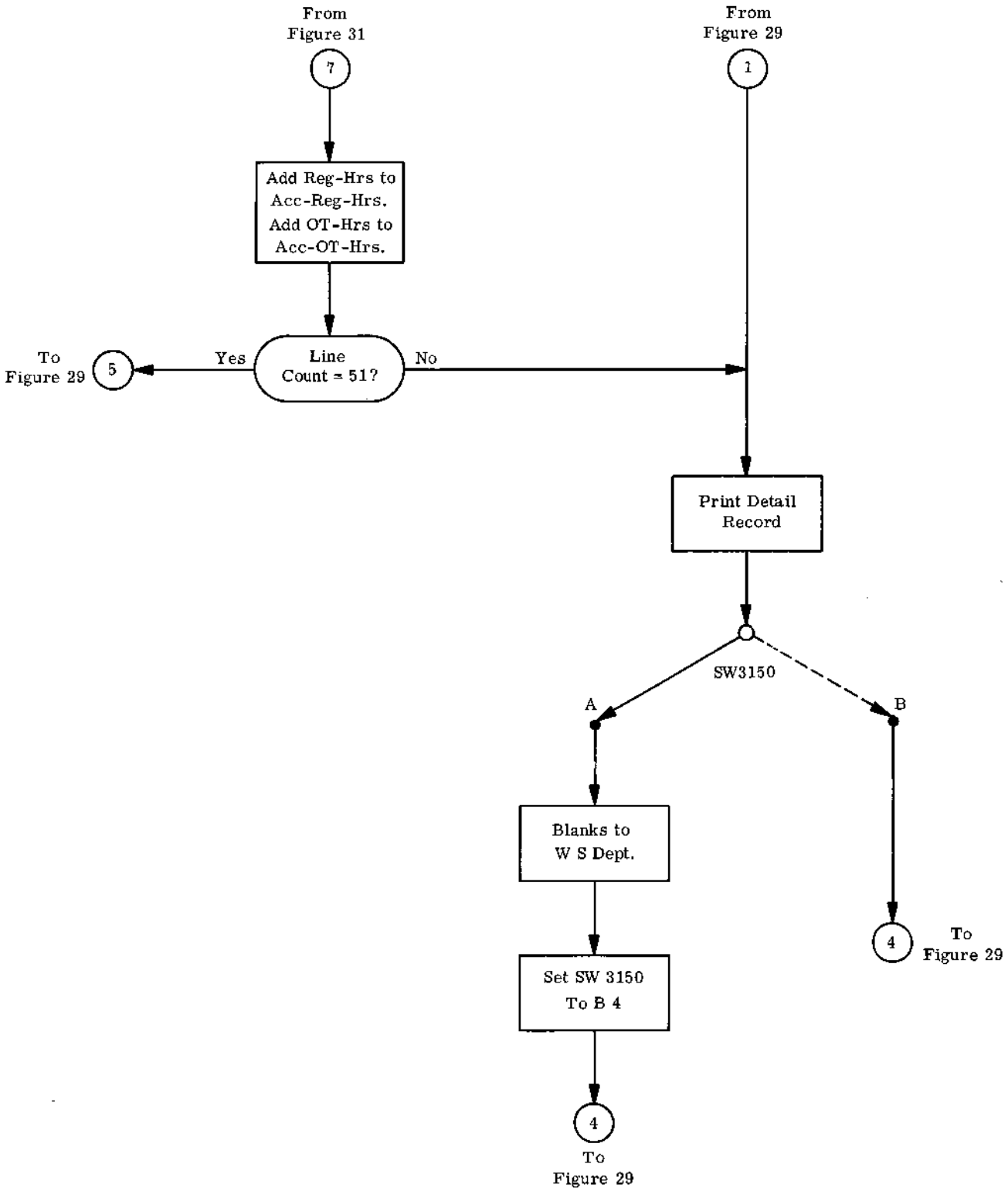


Figure 30. Job Ticket Summary Flow Chart (continued)

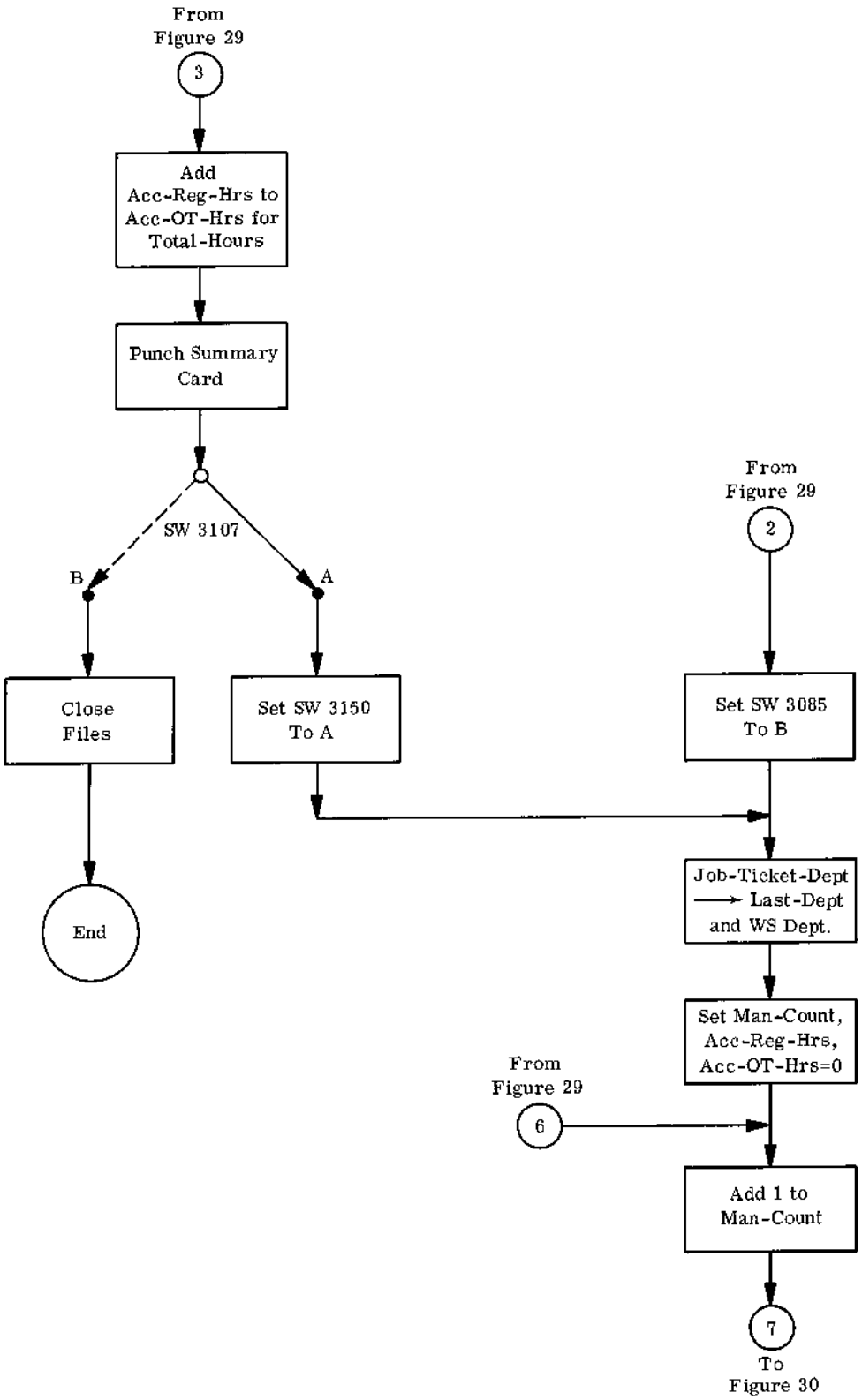


Figure 31. Job Ticket Summary Flow Chart (continued)

PREPARING THE SOURCE PROGRAM

Ordinarily, at this point the programmer would still have a large part of his work yet to be done -- the actual coding of the program. Using GECOM, he finds that, now, the problem is more than 75% complete. Although more space is devoted to this phase of the application than all preceding phases, what remains to be done is little more than translating the flow chart in Figures 29, 30, and 31 into about 40 English-language statements and defining the data involved in the program.

While not first in the GECOM compilation order, the Data Division of the source program is usually the first to be prepared.

DATA DIVISION PREPARATION

The General Compiler depends upon the information provided in the Data Division to define the data to be processed. Normally, it may be the responsibility of the programmer to organize this data logically. In the sample problem, this has already been done. The input job ticket cards collectively make up an input Job File. Each job ticket card is a Job Ticket Record. Each separate unit of data within the record is a Field of that record.

Figure 32 shows the completed Data Division form for the sample problem.

Although the illustration contains many typical entries, it should not be construed that these are the only types of entries possible. Appendix 3, which shows the complete source program compilation order, lists all optional and mandatory sections of the Data Division. Figure 32 of the sample problem contains all mandatory sections and entries except the END PROGRAM entry that must follow the last entry of the last section of the Data Division.

Data descriptions must be entered into the system in proper sequence, because sequence governs internal storage position. The Data Division form provides for sequence numbers in columns 1 through 6. Upon completion of all source program forms, they are punched into cards for system entry and the sequence numbers facilitate keeping the cards in proper order.

The Data Division is organized into sections and subsections which closely follow the logical organization of the data they represent. The mandatory subdivisions of the Data Division follow this arrangement:

```
DATA DIVISION
    FILE SECTION
        OUTPUT FILES
        INPUT FILES
    WORKING ~ STORAGE SECTION
END PROGRAM
```

Optional sections, such as the ARRAY, TRUE ~ FALSE, and INTEGER sections (as required by the individual program), precede the FILE SECTION. Other optional sections, such as the COMMON ~ STORAGE and CONSTANT sections, would follow the WORKING ~ STORAGE section.

The following explanations of the entries for the sample problem are not comprehensive. Detailed instructions for Data Division preparation are contained in the GE-225 GECOM Language Specifications.

Explanations are keyed to the sequence numbers that appear in columns 1 through 6 of Figure 32.

4000 DATA DIVISION.

This is the first entry of the Data Division and must always be included in the columns indicated and followed by a period. The sequence numbering system is at the programmer's option.

4005 FILE SECTION.

This entry precedes all file description entries. Columns used and the period are mandatory.

4010 OUTPUT FILES.

This entry must always follow the FILE SECTION. It indicates that the following descriptive entries are all part of the output files. Columns used and the period are mandatory.

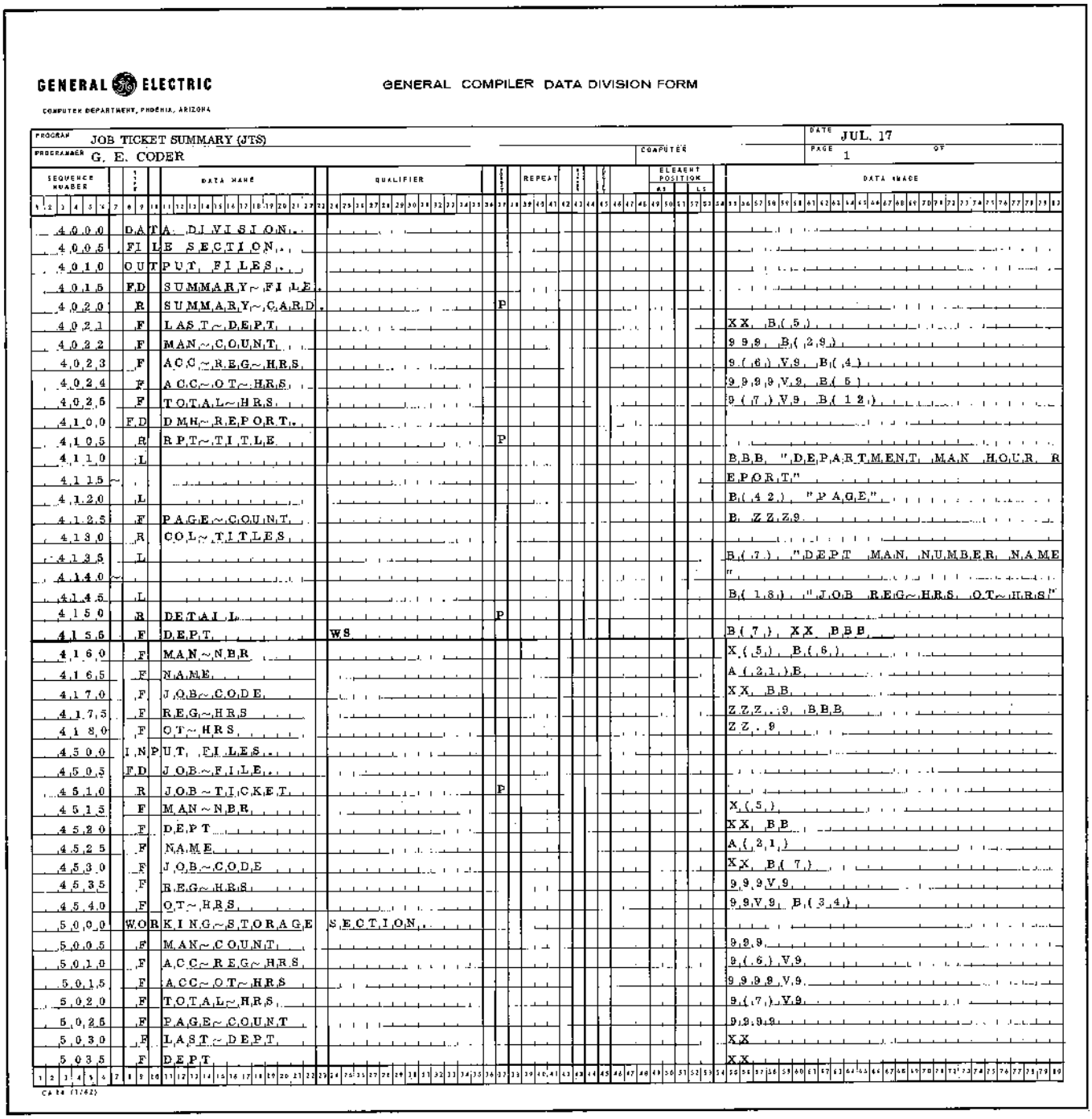
4015 FD SUMMARY~FILE.

This entry is typical of file name entries. The FD in columns 8 and 9 identify the entry as a File Description. Subsequent entries will describe the records that appear in this file. Any legal data name can appear in columns 11 through 22 as a file name. A following period is mandatory. When necessary a FD entry can be a sentence describing the mode of outgoing data (and incoming data if the FD is for INPUT files), the size of record blocks (if records are in blocks), and/or information pertaining to label records and control-keys. For example:

```
FD MASTER~FILE, RECORDING MODE IS BI-
NARY, BLOCK CONTAINS 500 WORDS,
CONTROL~KEY IS INDICATOR.
```

4020 R SUMMARY~CARD.

Record names are identified by an R in column 9. Record names must appear in columns 11 through 22 and be followed by a period. Any legal data name can be used to identify records. The P in column 37 (Format) indicates that all fields in this record are to be packed except for binary numerics. Packed



GENERAL ELECTRIC

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

GENERAL COMPILER DATA DIVISION FORM

PROGRAM: JOB TICKET SUMMARY (JTS)

PROGRAMMER: G. E. CODER

COMPUTER:

DATE: JUL. 17

PAGE: 1 OF

SEQUENCE NUMBER	TYPE	DATA NAME	QUALIFIER	SIGN	REPEAT	BLANK WHEN ZERO	JUST.	ELEMENT POSITION MS	ELEMENT POSITION LS	DATA IMAGE
4 0 0 0	D	A T A ~ D I V I S I O N .								
4 0 0 5	FI	L E ~ S E C T I O N .								
4 0 1 0	O U	T P U T ~ F I L E S .								
4 0 1 5	F D	S U M M A R Y ~ F I L E .								
4 0 2 0	R	S U M M A R Y ~ C A R D .		P						
4 0 2 1	F	L A S T ~ D E P T								X X B (5)
4 0 2 2	F	M A N ~ C O U N T								9 9 9 B (2 9)
4 0 2 3	F	A C C ~ R E G ~ H R S								9 (6) V 9 B (4)
4 0 2 4	F	A C C ~ O T ~ H R S								9 9 9 9 V 9 B (5)
4 0 2 5	F	T O T A L ~ H R S								9 (7) V 9 B (1 2)
4 1 0 0	F D	D M H ~ R E P O R T .								
4 1 0 5	R	R P T ~ T I T L E		P						
4 1 1 0	L									B B B " D E P A R T M E N T M A N H O U R R
4 1 1 5										E P O R T "
4 1 2 0	L									B (4 2) " P A G E "
4 1 2 5	F	P A G E ~ C O U N T								B Z Z Z 9
4 1 3 0	R	C O L ~ T I T L E S								
4 1 3 5	L									B (7) " D E P T M A N N U M B E R N A M E
4 1 4 0										"
4 1 4 5	L									B (1 8) " J O B R E G ~ H R S O T ~ H R S "
4 1 5 0	R	D E T A I L		P						
4 1 5 5	F	D E P T	W S							B (7) X X B B B
4 1 6 0	F	M A N ~ N B R								X (5) B (6)
4 1 6 5	F	N A M E								A (2 1) B
4 1 7 0	F	J O B ~ C O D E								X X B B
4 1 7 5	F	R E G ~ H R S								Z Z Z . 9 B B B
4 1 8 0	F	O T ~ H R S								Z Z . 9
4 5 0 0	I N P	U T ~ F I L E S .								
4 5 0 5	F D	J O B ~ F I L E .								
4 5 1 0	R	J O B ~ T I C K E T		P						
4 5 1 5	F	M A N ~ N B R								X (5)
4 5 2 0	F	D E P T								X X B B
4 5 2 5	F	N A M E								A (2 1)
4 5 3 0	F	J O B ~ C O D E								X X B (7)
4 5 3 5	F	R E G ~ H R S								9 9 9 V 9
4 5 4 0	F	O T ~ H R S								9 9 V 9 B (3 4)
5 0 0 0	W O R	K I N G ~ S T O R A G E	S E C T I O N .							
5 0 0 5	F	M A N ~ C O U N T								9 9 9
5 0 1 0	F	A C C ~ R E G ~ H R S								9 (6) V 9
5 0 1 5	F	A C C ~ O T ~ H R S								9 9 9 9 V 9
5 0 2 0	F	T O T A L ~ H R S								9 (7) V 9
5 0 2 5	F	P A G E ~ C O U N T								9 9 9 9
5 0 3 0	F	L A S T ~ D E P T								X X
5 0 3 5	F	D E P T								X X

CARD (1/62)

Figure 32. Job Ticket Summary Data Division

data is data that is entered into the GE-225 without regard to the GE-225 word length. Other possible entries in the Format column are described in the language specifications.

4021 F LAST~DEPT XX B(5)

The F designates the entry to be a field of whatever record entry precedes it. LAST~DEPT is a legal data name assigned for ready reference. The data image of the field is described in columns 55 through 80. Common symbols permitted here and their meanings are:

Symbol	Meaning
X	The position contains any of the GE-225 set of legal characters.
B	The position contains a leading or trailing blank; not a significant character of the field.
0	The position contains a leading or trailing zero; not a significant character of the field.
+	This position of the actual data will contain a plus sign (+) when the field is positive or a minus sign (-) when the field is negative.
-	This position of the actual data will contain a minus sign (-) when the field is negative or a blank when the field is positive.
A	The position contains alphabetic characters only, A through Z, or a blank.
9	The position contains integers only, 0 through 9.
V	Indicates an assumed decimal point. Neither the V nor the decimal point actually occupy a position in the field.
E	Separates the mantissa from the characteristic of a floating point number. The number following the E is the power of ten to which the number preceding the E must be raised. The E does not occupy an actual field position.
Z	An output entry only for editing. If this position is zero in the data field, the Z causes it to print as a blank.

Other data image symbols are defined in the GECOM language specifications.

Data images can be abbreviated, where convenient, by using parentheses and integers to show how many times a symbol is repeated. For example, the data images in the left column below are the equivalent of the data images in the right column:

A(7)9(5)	=	AAAAAAA99999
9(3)V9	=	999V9
A(4) B(3)	=	AAAA BBB

Thus, the data image entry for item 4021, XX B(5), means that the SUMMARY~CARD, when punched, will contain two alphanumeric characters (defined by XX) followed by five blanks (B(5)) before the next output field, if any. Refer to Figure 26, the sample Job Ticket Summary Card.

4110 L and
4115~

These entries are better explained together. The L in column 9, used only for output entries, indicates a literal. Literals are used to generate characters for headings. No data name is required in columns 11 through 22. The data image:

BBB "DEPARTMENT MAN HOUR R

entry of 4110 means that, on the output DMH~REPORT (Department Man Hour Report), to be produced on the high-speed printer, the first reading will start three spaces from the left (BBB) and will actually consist of the characters contained within the quotation marks, DEPARTMENT MAN HOUR R. . . . However, in this case, the desired heading was too long to be contained in the 26 columns allotted to the data image and a second entry is required, 4115. Entry 4115 contains ~ in column 7, indicating a carry-over from the preceding entry. The only other information required is the balance of the continued data image, EPORT". The result of these two entries is the first heading on the report illustrated in Figure 27.

Only a few entries have been explained. A study of all input and output file entries and a comparison with the input card and output card and report samples in Figures 25 through 27 will provide additional insight into Data Division preparation.

ENVIRONMENT DIVISION PREPARATION

Although the Environment Division itself is mandatory, the four sentence headings within it are optional and depend, not upon GECOM, but upon the source program.

The OBJECT~COMPUTER SENTENCE, which describes the computer for processing the object program, is included primarily for documentation and to override the automatic assignment of the object program to the first magnetic tape handler on the system, when desired.

The I~O~CONTROL sentence is used only if non-standard label-checking rerun information and/or multifile magnetic tapes are required.

The FILE~CONTROL sentence is used when the source program requires the identification and/or assignment of input/output files or hardware units. If the source program does not process input/output data, the FILE~CONTROL sentence can be omitted.

The COMPUTATION~MODE sentence is used when it is desired to perform computations on data in floating point format using floating point arithmetic.

For the Job Ticket Summary problem, the Environment Division would be prepared as shown in Figure 33.

The General Compiler Sentence Form is used; heading information, such as program and programmer identification are discretionary. Actual line entries must adhere to the rules detailed in the GE-225 GECOM Language Specifications. Some of these rules are mentioned in the line entry explanations that follow.

2000 ENVIRONMENT DIVISION.

The division heading is always the first entry for the division. The heading should begin in column 8 (recommended) or may be indented any number of spaces to the right. The heading must be followed by a period and no other information should follow on that line.

2005 and 2010 OBJECT~COMPUTER.

If this sentence is used, the sentence name should be started in column 8 and followed by a period. The sentence can start on the same line as the sentence name. In Figure 33, the compiler interprets the sentence to mean that the object program is to be performed on a GE-225 system with a 8192 word memory (2 MODULES) and the object program is to be input via card reader. To accomplish this, the General Compiler must produce the object program on punched cards via the card punch. Note that the sentence was too long to be completed on one line and was carried over to line 2010 and indented for clarity.

2015 FILE~CONTROL.

Like other sentence names, this one begins in column 8 as recommended. The first sentence is begun immediately after the name (with a blank between) and terminated with a period. All subsequent sentences must begin on a new line. The 2015 sentence in Figure 33 assigns the JOB ~ FILE (input) to the card reader buffer. The General Compiler interprets this to mean that data input through the card reader is to be treated as job file data.

2020 SELECT SUMMARY~FILE. . . .

This sentence assigns the SUMMARY~FILE to the card punch for output.

2025 SELECT DMH~REPORT. . . .

This sentence assigns the DMH REPORT to the high-speed printer for output. The DMH REPORT is considered as an output file and is therefore assigned to a peripheral like all files in the FILE~CONTROL Section.

PROCEDURE DIVISION PREPARATION

Once the programmer has flow charted the procedure to be followed and has defined all input and output data, it becomes relatively easy to state the processing steps to be followed in producing the desired output.

The programmer, having developed a working knowledge of GECOM language elements (verbs, names, constants, expressions, etc.) and their effects upon the object program, is prepared to document the procedure. Figure 34 illustrates the completed General Compiler Sentence Form for the Procedure Division of the Job Ticket Summary Problem. By relating the individual procedure statements and their explanations below to the flow charts in Figures 29 through 31, the overall procedure is more readily understood.

3000 PROCEDURE DIVISION.

Invariably the first entry for this division (and others) is the division name. It must be entered starting (preferably) in column 8 and terminated with a period.

3001 GO. . .

This opening sentence immediately and unconditionally transfers operation to the sentence identified by the sentence name, S3055.

3005 WPH SECTION.

This statement indicates that all procedure statements that follow are to be considered part of the WPH (Write Printer Heading) section until an END SECTION is encountered.

3010 through 3045

These statements comprise the WPH section which functions to advance the high-speed printer paper to the top of the page (3015), count pages (3020), space paper to the first print position (3025), print out the report title as defined by the literal entry at 4110 of the Data Division (3030), space paper to the next print line (3035), print out the column titles defined at 4135 through 4145 (3040),

GENERAL ELECTRIC

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

GENERAL COMPILER SENTENCE FORM

PROGRAM: JOB TICKET SUMMARY (JTS) | DATE: JUL. 17

PROGRAMMER: G. E. CODER | COMPUTER: GE-225 | PAGE:

SEQUENCE NUMBER	
2000.	ENVIRONMENT DIVISION.
2005	OBJECT~COMPUTER. 225, MEMORY SIZE 2 MODULES, ASSIGN OBJECT~PROGRAM
2010	TO CARD READER.
2015	FILE~CONTROL. SELECT JOB~FILE ASSIGN TO CARD READER BUFFER
2020	SELECT SUMMARY~FILE ASSIGN TO CARD PUNCH BUFFER.
2025	SELECT DMH~REPORT ASSIGN TO HSP ON PLUG 6.

GA 33 (10/61)

Figure 33. Job Ticket Summary Environment Division

GENERAL ELECTRIC

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

GENERAL COMPILER SENTENCE FORM

PROGRAM: JOB TICKET SUMMARY (JTS) — DATE: JUL. 17

PROGRAMMER: G. E. CODER — COMPUTER: GE-225 — PAGE:

SEQUENCE NUMBER		
3000	PROCEDURE DIVISION.	
3001	GO TO S3055.	
3005	WPH SECTION.	
3010	BEGIN.	
3015	ADVANCE DMH~REPORT TO TOP OF PAGE.	
3020	ADD 1 TO PAGE~COUNT.	
3025	ADVANCE DMH~REPORT 4 LINES.	
3030	WRITE RPT~TITLE.	
3035	ADVANCE DMH~REPORT 3 LINES.	
3040	WRITE COL~TITLES.	
3045	ADVANCE DMH~REPORT 2 LINES.	
3050	END WPH SECTION.	
S3055	5. OPEN ALL FILES.	
3060	MOVE 0 TO PAGE COUNT.	
3065	PERFORM WPH SECTION.	
3070	MOVE "ZZ" TO LAST~DEPT.	
S3075	5. READ JOB~FILE RECORD, IF END FILE GO TO S3180.	
3080	IF DEPT OF JOB~TICKET EQUALS LAST~DEPT GO TO S3125.	
S3085	S.W3085. GO TO S3090.	
S3090	0. ALTER S.W3085 TO PROCEED TO S3100.	
3095	GO TO S3115.	
S3100	0. TOTAL~HRS = ACC~REG~HRS + ACC~OT~HRS.	
3105	WRITE SUMMARY~CARD.	
S3107	SW3107. GO TO S3110.	
S3110	0. ALTER SW3150 TO PROCEED TO S3155.	
S3115	5. MOVE DEPT OF JOB~TICKET TO LAST~DEPT. DEPT OF WS.	
3120	MAN~COUNT = ACC~REG~HRS = ACC~OT~HRS = 0.	
S3125	5. ADD 1 TO MAN~COUNT.	
3130	ADD REG~HRS TO ACC~REG~HRS.	
3135	ADD OT~HRS TO ACC~OT~HRS.	
3140	IF LINE~COUNT EQUALS 51 GO TO S3170.	
S3145	5. WRITE DETAIL RECORD.	
S3150	SW3150. GO TO S3155.	
S3155	5. MOVE SPACES TO DEPT OF WS.	
3160	ALTER SW3150 TO PROCEED TO S3075.	
3165	GO TO S3075.	
S3170	0. PERFORM WPH SECTION.	
3175	GO TO S3145.	
S3180	0. ALTER SW3107 TO PROCEED TO S3182.	
3181	GO TO S3100.	
S3182	2. CLOSE JOB~FILE, SUMMARY FILE.	
3185	STOP RUN. "JTS."	

Figure 34. Job Ticket Summary Procedure Division

and space paper to the position for the first report entry (3045). This section will be referenced every time it becomes necessary to skip the paper to a new page after the preceding page has been printed with the desired number of lines.

3050 END WPH SECTION.

This entry signifies the end of the section and transfers control back to the procedure location following that which initiated performance of the WPH section (3070 or 3175).

3055 S3055. OPEN. . . .

S3055 is the data name assigned to this procedure statement to permit reference or transfer to the statement from other locations in the procedure. OPEN ALL FILES initiates the processing of all input and output files. No file can be read or written unless an OPEN sentence precedes processing. OPEN creates an input routine, places the first physical record in memory, sets up buffering if required, performs label checks on initial input tapes, and writes labels on initial output tapes. This one verb may generate hundreds of instructions.

3060 MOVE. . . .

This statement zeros the area reserved in memory for page counting, as assigned by Data Division entry 5025. Note the use of the zero constant. The MOVE verb fills the referenced field no matter how large.

3065 PERFORM. . . .

This statement transfers operation to the WPH SECTION at 3005. Upon completion of that section, control automatically reverts to the statefollowing the PERFORM, sequence number 3070. PERFORM thus enables repeated use of the same set of sentences in a section from various points in a program, where the function that the section performs is useful more than once. Coding time of the programmer is saved and duplicated use of memory is avoided.

3070 MOVE. . . .

This statement fills with Z's the memory locations reserved for LAST~DEPT. LAST~DEPT was defined by sequence number 5030 in the DATA DIVISION. In this case, MOVE forces an arbitrary NO answer to the test for change of DEPT~NO at 3080 for the very first record in order to allow various working storage areas to be cleared and to bypass initially punching a summary card, which would be meaningless.

3075 S3075. READ. . . .

Causes the first input record and each succeeding record to be made available from the card reader to the program for processing. If last record card is read, control transfers to the closing sequence beginning with sequence number 3180. If the department number on the card is the same as that in LAST~DEPT, control transfers to 3125.

3085 SW 3085. GO. . . .

This statement has been named SW (for switch) 3085 for ready reference. The number following the GO statement (S3090) is a sentence name that can be changed or modified to make the statement (3085) a program switch. Initial setting of the switch is to cause a GO to 3090 to bypass punching a summary card when the run begins and after 3070 has occurred.

3090 S3090. ALTER. . . .

This statement modifies the GO statement located at SW3085 to transfer to S3100 the next time that SW3085 is processed. SW3085 remains set to GO to 3100 for the remainder of the run in order to punch a summary card on each change of DEPT~NO.

3095 GO TO. . . .

This statement unconditionally transfers control to S3115 to continue processing with sentences which apply to the regular events on the change of department relative to printing detail records.

3100 S3100. TOTAL~HRS=. . . .

Statement is named because the GO statement at 3085 which references S3100 after 3085 is changed by 3090. The sentence causes the contents of the memory locations designated as ACC~REG~HRS and ACC~OT~HRS to be added together and placed in the memory location designated as TOTAL~HRS.

3105 WRITE. . . .

This statement causes the memory locations that were set aside for the summary card data by Data Division file description entry 4015 to be punched into a card.

3107 SW 3107. GO TO. . . .

This statement is another program switch as described above for SW3085. SW3107 is normally set to GO to S3110, but when the last record card (or sentinel card) is detected at 3075, there is a final summary card to be punched, then the files must be closed. Sentence 3180 sets SW3107 to enter the CLOSE sentence after the final summary card is punched.

3110 S3110. ALTER. . . .

This statement sets SW3150 to proceed to S3155 the next time it is processed. SW3150 handles the group suppression of printing of DEPT~NO. When a new department is detected at 3080, it is necessary to print that department number from working storage, but immediately after, blanks are moved to that working storage field (part of the Detail Record) and the MOVE of blanks must be bypassed until the next new department is encountered.

3115 S3115. MOVE. . . .

This statement places the contents of the memory location assigned to hold the job ticket department number to the memory locations assigned to hold the last department number and the working storage department number. The LAST~DEPT is for comparison with the department of the current Job Ticket to determine a change of department at 3080, while the department of working storage is to provide the department number for the first printing of a detail record for a new department, and blanks afterward.

3120 MAN~COUNT=. . . .

This is an assignment statement that sets to zero the memory locations reserved for the named field.

3125 S3125. ADD. . . .

The man count memory location is increased by one.

3130 ADD. . . .

The two named fields are added and the result replaces the previous value of ACC~REG~HRS.

3135 ADD. . . .

The two named fields are added and the result replaces the previous value of ACC~OT~HRS.

3140 IF. . . .

The contents of the LINE~COUNT memory location are compared with the constant, 51. If they are equal, control transfers to procedure statement S3170; if they are not equal, the next statement in sequence is taken (3145). LINE~COUNT = 51 indicates that the last line of a printer page has been printed and a new page (and new headings) must be started.

3145 S3145. WRITE. . . .

The DETAIL RECORD, defined in Data Division statements 4150 through 4180, which includes DEPT, MAN~NBR, NAME, JOB~CODE, REG~HRS, and OT~HRS fields, is printed as a line by the high-speed printer.

3150 SW3150. GO TO. . . .

This is another program switch similar to SW3085 and SW3107. It governs whether the detail record print line contains an actual department number or blanks.

3155 S3155. MOVE. . . .

This statement replaces the contents of the working storage DEPT field with blanks.

3160 ALTER. . . .

This statement changes the object of the GO statement at SW3150 from S3155 to S3075 to bypass S3155 and 3160 until a new department is read.

3165 GO TO. . . .

This statement unconditionally transfers control to S3145.

3170 S3170. PERFORM. . . .

Like statement 3065, this sentence transfers control to the WPH SECTION beginning at 3005. Upon completion of this section, control automatically reverts to the next statement in sequence, 3175. This is used to head up a new page after the capacity of the preceding page has been filled by a department's records.

3175 GO TO. . . .

This statement unconditionally transfers control to S3145.

3180 S3180. ALTER. . . .

This statement changes the object of the GO statement at SW 3107 from S3110 to S3182, so that CLOSE will occur after the final summary card is punched.

3181 GO TO. . . .

This statement unconditionally transfers control to S3100 to compute the final summary card TOTAL ~HRS.

3182 S3182. CLOSE. . . .

This statement terminates processings of the JOB~FILE and the SUMMARY~FILE. The card counts for the card reader and the card punch are printed out on the console typewriter.

GENERAL ELECTRIC

COMPUTER DEPARTMENT, PHOENIX, ARIZONA

GENERAL COMPILER SENTENCE FORM

PROGRAM: JOB TICKET SUMMARY (JTS) | DATE: JULY 17

PROGRAMMER: G. E. CODER | COMPUTER: | PAGE:

SEQUENCE NUMBER			
1000	IDENTIFICATION DIVISION.		
1005	PROGRAM~ID.	JTS.	
1010	AUTHOR.	G.E. CODER.	
1015	DATE~COMPILED.	JUL 17.	
1020	INSTALLATION.	GE COMPUTER DEPARTMENT, IPC PHOENIX, ARIZONA.	
1025	REMARKS.	SAMPLE PROGRAM.	

CR 13 (10/61)

Figure 35. Job Ticket Summary Identification Division

3185 STOP RUN "JTS"

This statement is used to generate object program coding for halting processing. In the form used here, the results will be

1. Program halts
2. END is printed by the console typewriter.
3. The literal "JTS" is printed by the console typewriter.

IDENTIFICATION DIVISION PREPARATION

This division enables the programmer to label the source program and provide program identification in the output Edited List.

The Identification Division is prepared on the General Compiler Sentence Form, as illustrated in Figure 35.

Entries for the Job Ticket Summary problem are explained:

1000 IDENTIFICATION DIVISION.

This mandatory heading indicates that entries following are for program identification only. The name should begin in column 8 and be followed by a period.

1005 PROGRAM~ID. JTS.

This entry is mandatory; the name, PROGRAM~ID, should appear beginning in column 8 and followed by a period. The actual program name, JTS, can consist of up to nine typewriter characters followed by a blank, a comma, or a period and can be indented any number of spaces. This name will appear as part of the heading of each page of the Edited List.

1010 AUTHOR. GE CODER

This entry is optional. If used, the sentence name should start in column 8 and be followed by a period. The sentence can be indented as desired, contain up to 30 BCD characters, and ended with a period. If provided, the author's name appears on each page of the Edited List.

1015 DATE COMPILED. JUL. 17

This entry is optional. It can contain up to 30 characters followed by a period. If provided, the compilation date appears on each page of the Edited List.

1020 INSTALLATION. . . .
1025 REMARKS. . . . .

These two sentences, as well as a NEXT~PROGRAM and a SECURITY sentence, are optional. If used, they can contain any information that the programmer wants to appear in the Edited List.

The Identification Division has no effect upon the compilation of the object program, other than that of appearing in the Edited List as described.

PRODUCING THE OBJECT PROGRAM

Upon completion of the GECOM forms for the source program, the data forms are transcribed to standard punched cards to form the source program deck and organized as shown in Figure 36.

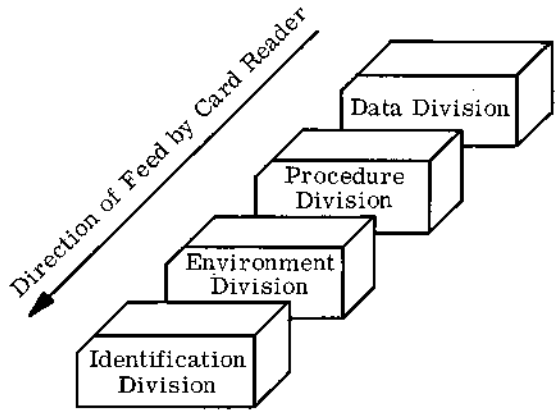


Figure 36. Source Program Deck Organization

A special GECOM call deck is placed before the source program deck and the cards are ready for input to the GE-225 via the card reader.

The minimum GE-225 system configuration for compiling the source program is:

GE-225 Central Processor (with 8192 words of core storage)
Console Typewriter
Card Reader
Card Punch
High-Speed Printer
Magnetic Tape Controller
Four Magnetic Tape Handlers
Five Magnetic Tape Handlers (optional)
Six Magnetic Tape Handlers (optional)

The GECOM Master Tape is mounted on the first magnetic tape handler on the system and includes a library of subroutines that might be required to complete the compiled object program. The source

program cards are loaded into the card reader. All console switches are set according to the desired Edited List output, all units are readied, and compilation is begun.

As compilation progresses, if errors have occurred during source program preparation, error messages and automatic halts will occur.

The GECOM Operations Manual contains detailed instructions for compilation of object programs, as well as descriptions of error stops and subroutines.

At the user's option, the source program card deck can be converted to magnetic tape prior to compilation. Compilation from magnetic tape is materially faster than from punched cards.

The outputs as a result of the GECOM compilation are:

1. The Object Program on punched cards or magnetic tape.

   a. If on cards, required subroutines can automatically be included or manually added from the external subroutine library.

   b. If on tape, required subroutines are automatically included at the end of the object program.

2. The Edited List, described later in detail.

3. Typewriter error messages.

4. A high-speed printer listing of all significant compilation process errors.

Upon successful completion of compilation, the object program is ready to be processed on the GE-225. The resulting outputs for the Job Ticket Summary problem would be a deck of punched cards like that illustrated in Figure 26, and a printed report as shown in Figure 27.

The Edited List for the Job Ticket Summary problem is shown, in abbreviated form, in Figures 37 through 46. The Edited List is an on-line printer listing that optionally provides full or partial documentation of the compiled program in a convenient and readily readable form. It facilitates source language error detection and correction and subsequent changes to the source and object programs, if required.

At the programmer's option, all or parts of the Edited List can be suppressed. Optional sections are indicated in the description that follows.

The first major section of the Edited List is the Source Listing which consists of a print-out of the Identification, Environment, Procedure, and Data Divisions as they appear in the source program. All programmer-prepared items are included; in addition, compiler-assigned sentence sequence numbers appear at the right side of the Procedure Division listing. Also, GAP symbols, as required by the source program, are printed between the "Sequence" and "Type" columns of the Data Division.

The second major section of the Edited List is the Reference Tables. This includes a "Procedure Name to GAP Symbol" list, identifying the compiler-assigned GAP symbols that are associated with each programmer-named source program sentence. This list is illustrated in Figure 40.

Three-character symbolic names of all subroutines required to complete the object program are also listed. Required subroutines optionally can be punched in cards as part of the object program if the object program is assigned to cards; if the object program is assigned to magnetic tape, subroutines are automatically included. At the programmer's option, he can elect to suppress all of the Edited List except the subroutine listing. The complete subroutine list is shown in Figure 40.

Sixteen subroutines are required to complete the object program for the Job Ticket Summary problem. Each subroutine contains a large number of machine instructions, and represents several weeks of original programming effort. Complete libraries of subroutines are available for use with GECOM, and provide appreciable savings in user programming effort.

As a trouble-shooting aid, the compiler-assigned GAP symbols and their memory addresses in octal code are also listed. Although only a portion of the list is shown in Figure 40, 241 symbols and addresses were generated in the compilation of the Job Ticket Summary problem.

The third major section of the Edited List is the Object Listing. The first print-out in this section lists the Procedure Division sentences and the coding produced by each sentence during compilation. This list is shown in its entirety in Figures 41 through 45. Each line of coding represents one instruction, with the octal machine coding shown on the left (second column) and the GAP symbolic coding on the right.

The next print-out in the Object Listing section is the "Storage Reservations and Procedure Constants." This includes, in GAP format, constants derived from the Data Division, a list of memory locations reserved by the General Compiler for specific GAP symbols, and constants derived from the Procedure Division in GAP format. Figure 45 shows part of this list.

The Object Listing includes an "Input/Output Coding" print-out showing all input/output file tables, control coding, and service routines. A complete listing of this subsection for the sample problem requires 439 line entries. Part of the Input/Output Coding list is shown in Figure 46.

The final print-out of the Object Listing and the Edited List is "Location Assignments for GECOM Common Constants," Figure 46. This print-out contains the memory locations for object program constants and the compiler-assigned symbols for the constants. For the sample problem, the complete constant listing contains 138 entries.

```
      GECOM LISTING OF JTS                                          PAGE 001

                GE CODER                                JUL 17

      S O U R C E   L I S T I N G

1000  IDENTIFICATION DIVISION.

1005  PROGRAM_ID.          JTS.
1010  AUTHOR.              GE CODER.
1015  DATE_COMPILED.       JUL 17.
1020  INSTALLATION.        GE COMPUTER DEPARTMENT IPC PHOENIX ARIZONA.
1025  REMARKS.             SAMPLE PROBLEM

2000  ENVIRONMENT DIVISION.

2005       OBJECT_COMPUTER, 225, MEMORY SIZE 2 MODULES, ASSIGN OBJECT_PROGRAM
2010           TO CARD READER.
2015       FILE_CONTROL, SELECT JOB_FILE ASSIGN TO CARD READER BUFFER,
2020                     SELECT SUMMARY_FILE ASSIGN TO CARD PUNCH BUFFER,
2025                     SELECT DMH_REPORT ASSIGN TO HSP ON PLUG 6.

3000  PROCEDURE DIVISION.

3001            GO TO S3055                                          0010
3005  WPH SECTION.                                                   0020
3010  BEGIN.                                                         0030
3015            ADVANCE DMH_REPORT TO TOP OF PAGE.                   0040
3020            ADD 1 TO PAGE_COUNT.                                 0050
3025            ADVANCE DMH_REPORT 4 LINES.                          0060
3030            WRITE RPT_TITLE.                                     0070
3035            ADVANCE DMH_REPORT 3 LINES.                          0080
3040            WRITE COL_TITLES.                                    0090
3045            ADVANCE DMH_REPORT 2 LINES.                          0100
3050  END WPH SECTION.                                               0110
3055  S3055.    OPEN ALL FILES.                                      0120
3060            MOVE 0 TO PAGE_COUNT.                                0130
3065            PERFORM WPH SECTION.                                 0140
3070            MOVE #ZZ# TO LAST_DEPT.                              0150
3075  S3075.    READ JOB_FILE RECORD IF END FILE GO TO S3180.        0160
3080            IF DEPT OF JOB_TICKET EQUALS LAST_DEPT GO TO S3125.  0170
3085  SW3085.   GO TO S3090.                                         0180
3090  S3090.    ALTER SW3085 TO PROCEED TO S3100.                    0190
3095            GO TO S3115.                                         0200
3100  S3100.    TOTAL_HRS  =  ACC_REG_HRS + ACC_OT_HRS.             0210
3105            WRITE SUMMARY CARD.                                  0220
3107  SW3107.   GO TO S3110.                                         0230
3110  S3110.    ALTER SW3150 TO PROCEED TO S3155.                    0240
3115  S3115     MOVE DEPT OF JOB_TICKET TO LAST_DEPT, DEPT OF WS.    0250
3120            MAN_COUNT = ACC_REG_HRS = ACC_OT_HRS = 0.            0260
3125  S3125.    ADD 1 TO MAN_COUNT.                                  0270
3130            ADD REG_HRS TO ACC_REG_HRS.                          0280
3135            ADD OT_HRS TO ACC_OT_HRS.                            0290
3140            IF LINE_COUNT EQUALS 51 GO TO S3170.                 0300
```

Figure 37. Edited List

```
      GECOM LISTING OF JTS                                          PAGE 002

                GE CODER                                    JUL 17

S O U R C E   L I S T I N G   ( C O N T. )

3145  S3145.  WRITE DETAIL RECORD.                                          0310
3150  SW3150. GO TO S3155.                                                  0320
3155  S3155.  MOVE SPACES TO DEPT OF WS.                                    0330
3160          ALTER SW3150 TO PROCEED TO S3075.                             0340
3165          GO TO S3075.                                                  0350
3170  S3170.  PERFORM WPH SECTION.                                          0360
3175          GO TO S3145.                                                  0370
3180  S3180.  ALTER SW3107 TO PROCEED TO S3182.                             0380
3181          GO TO S3100.                                                  0390
3182  S3182.  CLOSE JOB_FILE, SUMMARY_FILE.                                 0400
3185          STOP RUN #JTS#.                                               0410


4000  DATA DIVISION.


(SEQ  GAP T DATA NAME    QUALIFIER     F  RPT B J E MS LS DATA IMAGE)

4005  FILE SECTION
4010  OUTPUT FILES.
4015  000FD SUMMARY_FILE.
4020  000 R SUMMARY_CARD              P
4021      F LAST_DEPT                                   XX B(5)
4022      F MAN_COUNT                                   999 B(29)
4023      F ACC_REG_HRS                                 9(6)V9 B(4)
4024      F ACC_OT_HRS                                  9999V9 B(5)
4025      F TOTAL_HRS                                   9(7)V9 B(12)
4100  001FD DMH_REPORT.
4105  000 R RPT_TITLE                 P
4110      L                                             BBB #DEPARTMENT MAN HOUR R
4115                                                    EPORT#
4120    - L                                             B(42) #PAGE#
4125      F PAGE_COUNT                                  B ZZZ9
4130  001 R COL_TITLES
4135      L                                             B(7) #DEPT MAN NUMBER NAME
4140                                                    #
4145    - L                                             B(18) #JOB REG-HRS OT-HRS#
4150  002 R DETAIL                    P
4155      F DEPT          WS                            B(7) XX BBB
4160      F MAN_NBR                                     X(5) B(6)
4165      F NAME                                        A(21)B
4170      F JOB_CODE                                    XX BB
4175      F REG_HRS                                     ZZZ.9 BBB
4180      F OT_HRS                                      ZZ.9
4500  INPUT FILES.
4505  002FD JOB_FILE.
4510  000 R JOB_TICKET                P
4515      F MAN_NBR                                     X(5)
4520  00J F DEPT                                        XX BB
4525      F NAME                                        A(21)
4530      F JOB_CODE                                    XX B(7)
4535  05A F REG_HRS                                     999V9
```

Figure 38. Edited List

```
      GECOM LISTING OF JTS                                          PAGE 003

                  GE CODER                                  JUL 17

      S O U R C E   L I S T I N G   ( C O N T. )

(SEQ  GAP T DATA NAME      QUALIFIER    F RPT B J E MS LS  DATA IMAGE)

4540  06A F OT_HRS                                        99V9 B(34)
5000  WORKING_STORAGE SECTION.
5005  04A F MAN_COUNT                                     999
5010  02A F ACC_REG_HRS                                   9(6)V9
5015  03A F ACC_OT_HRS                                    9999V9
5020  01A F TOTAL_HRS                                     9(7)V9
5025  00A F PAGE_COUNT                                    9999
5030  01J F LAST_DEPT                                     XX
5035  02J F DEPT                                          XX

      END PROGRAM.
```

Figure 39. Edited List

```
GECOM LISTING OF JTS                                          PAGE 004

                                                              JUL 17
          GE CODER

R E F E R E N C E   T A B L E S

PROCEDURE NAME TO GAP SYMBOL

  (GAP  PROCEDURE NAME)

   A01  S3055
   A03  S3075
   A07  S3090
   A08  S3100
   A11  S3110
   A09  S3115
   A05  S3125
   A15  S3145
   A13  S3155
   A14  S3170
   A04  S3180
   A16  S3182
   A06  SW3085
   A10  SW3107
   A12  SW3150
   A02  WPH


NAMES OF SUB-ROUTINES REQUIRED

  (GAP  SECTION NAME)

   ADV
   FLX
   FXP
   RCS
   RLC
   TYP
   ZAM
   ZBN
   ZCB
   ZED
   ZNB
   ZNN
   ZOT
   ZSC
   ZSG
   ZUA

GAP SYMBOLIC TO OCTAL LOCATION

  (GAP OCTAL    GAP OCTAL    GAP OCTAL    GAP OCTAL    GAP OCTAL    GAP OCTAL)

   00A 01363    00J 01402    00S 01110 00TCP 01713 00TXT 01712    00U 01646
   00V 01714 00W00 01664    00WE 01675    00W 01664    00X 01406    00Y 01406
 00Z00 02040    01A 01366    01J 01403    01S 01120 01TCP 02006 01TXT 02005
   01U 01737    01V 02007 01W00 02032 01W01 02034 01W02 02036  01WE 01772
   01W 01755    01X 01406 01Z00 02076 01Z01 02120 01Z02 02133    02A 01370
```

Figure 40. Edited List

```
GECOM LISTING OF JTS                                                PAGE 006

          GE CODER                                          JUL 17

     O B J E C T   L I S T I N G

3001          GO TO S3055.                                          0010
     01144  2601204              BRU   A01

3005 WPH SECTION.                                                   0020
3010 BEGIN.                                                         0030
     01145  1420001    A02       INX   1       1
     01146  0000001              LDA   1
     01147  2701203              STO   A02#/@

3015          ADVANCE DMH_REPORT TO TOP OF PAGE.                    0040
     01150  0721142              SPB   ADV     1
            01142    ADV         EQU   TV2-02
     01151  2000006              OCT   2000006
     01152  0000252              LDA   ZER
     01153  0301405              STA   PC6

3020          ADD 1 TO PAGE_COUNT.                                  0050
     01154  0001363              LDA   00A
     01155  0101442              ADD   0J0
     01156  0301363              STA   00A

3025          ADVANCE DMH_REPORT 4 LINES.                           0060
     01157  0001444              LDA   0J1
     01160  0721142              SPB   ADV     1
     01161  0000006              OCT   0000006
     01162  0001444              LDA   0J1
     01163  0101405              ADD   PC6
     01164  0301405              STA   PC6

3030          WRITE RPT_TITLE.                                      0070
     01165  0722032              SPB   01W00   1

3035          ADVANCE DMH_REPORT 3 LINES.                           0080
     01166  0001446              LDA   0J2
     01167  0721142              SPB   ADV     1
     01170  0000006              OCT   0000006
     01171  0001446              LDA   0J2
     01172  0101405              ADD   PC6
     01173  0301405              STA   PC6

3040          WRITE COL_TITLES.                                     0090
     01174  0722034              SPB   01W01   1

3045          ADVANCE DMH_REPORT 2 LINES.                           0100
```

Figure 41. Edited List

```
GECOM LISTING OF JTS                                                    PAGE 007

                GE CODER                                                JUL 17


     O B J E C T   L I S T I N G   ( C O N T. )

     01175  0001450              LDA   0J3
     01176  0721142              SPB   ADV     1
     01177  0000006              OCT   0000006
     01200  0001450              LDA   0J3
     01201  0101405              ADD   PC6
     01202  0301405              STA   PC6

                                                                        0110
3050 END WPH SECTION.

     01203  2601203   A02#/@     BRU   A02#/@

                                                                        0120
3055 S3055.  OPEN ALL FILES.

     01204  0721646   A01        SPB   00U     1
     01205  0721737              SPB   01U     1
     01206  0721461              SPB   02U     1

                                                                        0130
3060         MOVE 0 TO PAGE_COUNT.

     01207  0001452              LDA   0J4
     01210  0301363              STA   00A

                                                                        0140
3065         PERFORM WPH SECTION.

     01211  0721145              SPB   A02     1

                                                                        0150
3070         MOVE #ZZ# TO LAST_DEPT.

     01212  0001457              LDA   0A5
     01213  0301403              STA   01J

                                                                        0160
3075 S3075.  READ JOB_FILE RECORD IF END FILE GO TO S3180.

     01214  0001315   A03        LDA   A04
     01215  0001214              LDA   *-1
     01216  2701571              STO   02T
     01217  0721511              SPB   02W     1

                                                                        0170
3080         IF DEPT OF JOB_TICKET EQUALS LAST_DEPT GO TO S3125.

     01220  0001403              LDA   01J
     01221  2000314              EXT   EXB
     01222  0300654              STA   XYZ
     01223  0001402              LDA   00J
     01224  2000314              EXT   EXB
     01225  0200654              SUB   XYZ
     01226  2514002              BZE   A05
     01227  2601262

                                                                        0180
3085 SW3085, GO TO S3090.

     01230  2601231   A06        BRU   A07

                                                                        0190
3090 S3090.  ALTER SW3085 TO PROCEED TO S3100.
```

Figure 42. Edited List

```
GECOM LISTING OF JTS                                                    PAGE 008

          GE CODER                                                JUL 17

     O B J E C T   L I S T I N G   ( C O N T. )

     01231  0001235    A07      LDA   A08
     01232  0001231             LDA   *-1
     01233  2701230             STO   A06

3095          GO TO S3115.                                                  0200

     01234  2601251             BRU   A09

3100 S3100. TOTAL_HRS = ACC_REG_HRS + ACC_OT_HRS.                          0210

     01235  1001370    A08      DLD   02A
     01236  0721143             SPB   FXP      1
            01143    FXP      EQU   TV2-01
     01237  0101372             ZO1   03A
     01240  0023025             OCT   0023025
     01241  0721143             SPB   FXP      1
     01242  0300031             ZO3   025
     01243  1301366             DST   01A

3105          WRITE SUMMARY_CARD.                                           0220

     01244  0721664             SPB   00W00    1

3107 SW3107. GO TO S3110.                                                  0230

     01245  2601246    A10      BRU   A11

3110 S3110.  ALTER SW3150 TO PROCEED TO S3155.                             0240

     01246  0001305    A11      LDA   A13
     01247  0001246             LDA   *-1
     01250  2701304             STO   A12

3115 S3115.  MOVE DEPT OF JOB_TICKET TO LAST_DEPT, DEPT OF WS.             0250

     01251  0001402    A09      LDA   00J
     01252  0301403             STA   01J
     01253  0301404             STA   02J

3120          MAN_COUNT = ACC_REG_HRS = ACC_OT_HRS = 0.                    0260

     01254  1001452             DLD   0J4
     01255  1301372             DST   03A
     01256  2511002             SRD   002
     01257  1301370             DST   02A
     01260  2512202             SLD   002
     01261  0301374             STA   04A

3125 S3125.  ADD 1 TO MAN_COUNT.                                           0270

     01262  0001374    A05      LDA   04A
     01263  0101442             ADD   0J0
     01264  0301374             STA   04A

3130          ADD REG_HRS TO ACC_REG_HRS.                                  0280
```

Figure 43. Edited List

```
GECOM LISTING OF JTS                                          PAGE 009
                GE CODER                                      JUL 17

      O B J E C T   L I S T I N G   ( C O N T. )

      01265  1001370              DLD   02A
      01266  0721143              SPB   FXP     1
      01267  0101376              ADD   05A
      01270  0023025              OCT   0023025
      01271  0721143              SPB   FXP     1
      01272  0300025              STA   021
      01273  1301370              DST   02A

3135          ADD OT_HRS TO ACC_OT_HRS.                                0290

      01274  1001372              DLD   03A
      01275  1101400              DAD   06A
      01276  1301372              DST   03A

3140          IF LINE_COUNT EQUALS 51 GO TO S3170.                     0300

      01277  0001405              LDA   PC6
      01300  0201454              SUB   0J5
      01301  2514002              BZE   A14
      01302  2601313

3145 S3145.   WRITE DETAIL RECORD.                                     0310

      01303  0722036    A15       SPB   01W02   1

3150 SW3150.  GO TO S3155.                                            0320

      01304  2601305    A12       BRU   A13

3155 S3155.   MOVE SPACES TO DEPT OF WS.                              0330

      01305  0001460    A13       LDA   0A6
      01306  0301404              STA   02J

3160          ALTER SW3150 TO PROCEED TO S3075.                       0340

      01307  0001214              LDA   A03
      01310  0001307              LDA   *-1
      01311  2701304              STO   A12

3165          GO TO S3075.                                            0350

      01312  2601214              BRU   A03

3170 S3170.   PERFORM WPH SECTION.                                    0360

      01313  0721145    A14       SPB   A02     1

3175          GO TO S3145.                                            0370

      01314  2601303              BRU   A15

3180 S3180.   ALTER SW3107 TO PROCEED TO S3182.                       0380
```

Figure 44. Edited List

```
GECOM LISTING OF JTS                                                PAGE 010

          GE CODER                                          JUL 17

     O B J E C T   L I S T I N G   ( C O N T. )

     01315  0001321    A04      LDA   A16
     01316  0001315             LDA   *-1
     01317  2701245             STO   A10

3181          GO TO S3100.                                          0390

     01320  2601235             BRU   A08

3182 S3182.  CLOSE JOB_FILE, SUMMARY_FILE.                         0400

     01321  0721576    A16      SPB   02V      1
     01322  0721714             SPB   00V      1

3185          STOP RUN #JTS#.                                       0410

     01323  0761141             SPB   TYP      3
              01141    TYP      EQU   TV2-03
     01324  0254524             ALF   END
     01325  0761141             SPB   TYP      3
     01326  2000001             OCT   2000001
     01327  0001456             LDA   0A4
     01330  0761141             SPB   TYP      3
     01331  0373737             OCT   0373737
     01332  0721140             SPB   RLC      1
              81140    RLC      EQU   TV2-04


     STORAGE RESERVATIONS AND PROCEDURE CONSTANTS

     01333  0242547    0A0      ALF   DEP
     01334  0215163             ALF   ART
     01335  0442545             ALF   MEN
     01336  0636044             ALF   T M
     01337  0214560             ALF   AN
     01340  0304664             ALF   HOU
     01341  0516051             ALF   R R
     01342  0254746             ALF   EPO
     01343  0516300             ALF   RTO
     01344  0472127    0A1      ALF   PAG
     01345  0256060             ALF   E
     01346  0242547    0A2      ALF   DEP
     01347  0636044             ALF   T M
     01350  0214560             ALF   AN
     01351  0456444             ALF   NUM
     01352  0222551             ALF   BER
     01353  0604521             ALF    NA
     01354  0442500             ALF   MEO
     01355  0414622    0A3      ALF   JOB
     01356  0605125             ALF    RE
     01357  0274030             ALF   G-H
     01360  0516260             ALF   RS
     01361  0466340             ALF   OT-
     01362  0305162             ALF   HRS
```

Figure 45. Edited List

```
GECOM LISTING OF JTS                                          PAGE 011

          GE CODER                                            JUL 17

O B J E C T   L I S T I N G   ( C O N T. )
INPUT-OUTPUT CODING (Partial Listing)

          01100                        LOC     1100
01100     0000262    02S               ALF     02S
01101     0000010                      OCT     10
01102     2500200                      RCD     128
01103     2500400                      RCD     256
01104     2000001                      EXT     1
01105     0000000                      OCT     0
01106     0000000                      OCT     0
01107     0000000                      OCT     0
          01461                        ORG     BIN
01461     0001504    02U               LDA     02W-5


LOCATION ASSIGNMENTS FOR GECOM COMMON CONSTANTS (Partial Listing)
(ASSEMBLED IN FRONT OF PROCEDURE CODING)

          01144     TV2       BSS     0
          00572     IXY       EQU     378
          00252     ZER       EQU     170
          00252     Z00       EQU     ZER
          00254     Z01       EQU     172
          00255     Z02       EQU     173
          00256     Z03       EQU     174
          00257     Z04       EQU     175
          00260     Z05       EQU     176
          00261     Z06       EQU     177
          00262     Z07       EQU     178
          00263     Z08       EQU     179
          00264     Z09       EQU     180
          00265     Z10       EQU     181
          00266     Z11       EQU     182
          00267     Z12       EQU     183
          00270     Z17       EQU     184
          00271     Z18       EQU     185
          00272     Z19       EQU     186
          00273     Z20       EQU     187
          00274     Z24       EQU     188
          00275     Z25       EQU     189


END OF GECOM LISTING
```

Figure 46. Edited List

APPENDIX 1. THE GENERAL COMPILER VOCABULARY

Words and terms that appear in the following list must be considered to be part of the General Compiler vocabulary and must not be used by the systems programmer in forming data or procedure names, nor may they be used in any manner in a source program other than as provided by the GECOM Language Specifications.

Where warranted, many of the terms have been defined or explained. Terms not so explained were deemed to be self-evident in meaning. In addition, the body of the manual contains many examples that illustrate the use of most of the vocabulary terms.

ABS - Absolute value, or magnitude, of a number, regardless of sign.

ACCESS - Part of descriptive name Mass Random Access Data Storage.

ADD - To add two quantities and store the sum in either the last-named field or the specified field.

ADVANCE - To vertically skip or slew the printer paper.

AFTER

ALL

ALTER - To modify a sequence of operations specified in one or more GO sentences.

AND - A logical operator.

ARE

ARRAY - A multi-valued field that may be referenced by name and subscript. An array may be one, two, or three dimensional and may have corresponding number of subscripts. An array must be defined in the Array Section of the Data Division.

ASSIGN - To direct the placement of a file or program to an input-output media.

ASSIGNMENT - To evaluate an arithmetic expression and assign the result to a field. To equate data names.

ATAN - Are tangent. A mathematical function that may be used within arithmetic expressions. Calculated in floating point arithmetic.

AUTHOR - An optional Identification Division sentence name.

BEGIN - Entrance point to a source program section.

BEGINNING

BGN~FIL~LABL - A tape record preceding each file of a multi-file tape.

BGN~TAP~LABL - The first record on any tape except in multi-file tape.

BINARY - Pertaining to the binary number system, as opposed to decimal or binary coded decimal.

BLOCK - See Glossary

BUFFER - A device which stores data temporarily during transfer operations.

BY

CARD

CLOSE - To terminate processing of input or output reels and files with optional rewind and/or lock.

COMMON (~STORAGE) - An optional Data Division Section name.

COMPUTATION ~ MODE - An optional Environment Division sentence name.

CONSTANT - An optional Data Division section name.

CONTAINS

CONTROL - Interpretation and execution of operations.

CONTROL~KEY - The field or fields by which a record is identified.

COPY - To duplicate from another area.

COPYING

COS - Cosine. A mathematical function that may be used within arithmetic expressions. Calculated in floating point arithmetic.

DATA - A GECOM Division name.

DATE~COMPILED - An optional Identification Division sentence name.

DEPENDING

DIVIDE - To divide one number into another and store the result in the last-named field or the specified field.

DIVISION - A major section of a GECOM source program.

EIGHT (S) - A figurative constant used in procedure sentences.

END~FIL~LABL - A tape record that appears once after the last data record and tapemark of the last tape reel of a file. On multi-file tapes, this record appears after each file.

ENDING

END (PROGRAM) - A mandatory terminal entry for the Data Division.

END~TAP~LABL - A tape record that follows the last data record and tape mark on intermediate reels of a multi-reel file. Not used on multi-file tapes.

ENTER - To permit programmer insertion of General Assembly Program (GAP) coding into the GECOM source program. GAP coding must be terminated by END.

ENVIRONMENT - A GECOM Division name.

EQ - Equal to, Equals. Used in relational expressions.

EQUAL (S) - See above

ERROR

EVERY

EXCEEDS - Is greater than

EXCHANGE - To transpose the contents of two fields. Data images must be identical.

EXIT

EXP - Exponential. A mathematical function that may be used in arithmetic expressions. Calculated in floating point arithmetic.

FILE(S) - A set of records. A mandatory Data Division section name.

FILE~CONTROL - An optional Environment Division sentence name.

FIVE(S) - A figurative constant used in procedure sentences.

FLOATING POINT - An arithmetic mode of calculation.

FLPT - See above.

FOR

FOUR(S) - A figurative constant used in procedure sentences.

FROM

GIVING

GO - To leave the normal sequence of procedures.

GR - Exceeds or Greater than. Used in relational expressions.

GREATER - See above

GROUP - A set of data within a record and consisting of fields.

HARDWARE - Equivalent to "data processing equipment."

HIGH

IDENTIFICATION - A GECOM Division name.

IF [NOT] - To transfer control to the specified sentence if the stated condition is satisfied (true), or to the next sentence if the stated condition is not satisfied (false).

INPUT - A mandatory Data Division section name.

INSTALLATION - An optional Identification Division sentence name.

I~O~CONTROL - An optional Environment Division sentence name.

INTEGER - An optional Data Division section name.

INTO

IS

LABEL

LESS

LINE COUNT

LINES

LN - Natural logarithm. A mathematical function that may be used in arithmetic expressions. Calculated in floating-point arithmetic.

LOCK - To prevent a tape from being read or written by program control.

LOG - Common Logarithm. A mathematical function that may be used in arithmetic expressions. Calculated in floating point arithmetic.

LS - LESS than. Used in relational expressions.

MAGNETIC - Part of descriptive name, Magnetic Tape Handler.

MASS - Part of descriptive name, Mass Random Access Data storage.

MEMORY - Main storage, core storage.

MODE - A system of data presentation or processing within the information processing system.

MODULE(S) - Refers to core memory size; one module is 4096 words of storage.

MOVE - To transfer a constant, element, field group, record, or array to a constant, element, etc. of the same size.

MULTIPLE

MULTIPLY - To multiply two quantities and store the result in the last-named field or the specified field.

NEGATIVE

NEQ - Not equal to. Used in relational expressions.

NEXT~PROGRAM - An optional Identification Division sentence name.

NGR - Not Greater Than. Used in relational expressions.

NINE(S) - A figurative constant used in procedure sentences.

NLS - Not Less Than. Used in relational expressions.

NO

NOT - May be used in relational expressions. In logical expressions, it is an exclusive negative.

NOTE - To permit the programmer to write explanatory material in the source program for inclusion in the Edited List, but excluded from the compilation.

OBJECT~COMPUTER - An optional Environment Division sentence name.

OBJECT~PROGRAM - See Glossary

OF

OMITTED

ON

ONE(S) - A figurative constant used in procedure sentences.

OPEN - To initiate the processing of input and output files. Checks or writes labels and does other input-output functions.

OPTIONAL

OR - A logical operator

OUTPUT - A mandatory Data Division section name.

PAGE

PAPER - Pertaining to High-Speed Printer forms.

PERFORM - To cause the specified section to be executed. Control automatically reverts to sentence following the PERFORM.

PLUG(S) - Refers to connectors on the controller selector to which input-output unit controllers are attached.

POSITION

POSITIVE

PRINTER(S) - Pertaining to High-Speed Printer.

PROCEDURE - A GECOM Division name.

PROCEED

PROGRAM - A complete sequence of data processing instructions. May refer to an object program or a source program.

PROGRAM~ID - A mandatory Identification Division sentence name.

PROGRAMMED

PUNCH - Pertaining to Card Punch or punched cards.

RANDOM - Part of descriptive name Mass Random Access Data Storage.

READ - To allow entry of data from console switches. To allow entry of next record or group from an input file and transfer control to another sentence when END OF FILE is reached. To copy or advance an input tape file until a condition is met and specify next controlling sentence.

READER - Card Reader

RECORD(S) - A set of data within a file and consisting of fields.

RECORDING

REEL - Magnetic tape reel.

RELOCATABLE - As in Relocatable Section

REMARKS - An optional Identification Division sentence name.

REPLACING

RERUN

REWIND - As in rewind magnetic tape.

ROUNDED - To shorten a number by dropping least significant digits and adjusting the value of the remaining least significant digit by one if the last dropped number is 5 or more.

RUN - To process. One pass through an object program by the computer is a run.

SECTION - A series of consecutive statements in GECOM having a name for reference purposes.

SECURITY - An optional Identification Division sentence name.

SEE

SELECT

SEQUENCED

SEVEN(S) - A figurative constant used in procedure sentences.

SIN - Sine. A mathematical function that may be used within arithmetic expressions. Calculated in floating point arithmetic.

SIX - A figurative constant used in procedure sentences.

SIX(ES) - See SIX -

SIZE

SPACE(S) - A figurative constant used in procedure sentences.

SPEED

SQRT - Square Root. A mathematical function that may be used within arithmetic expressions. Calculated in floating point arithmetic.

STANDARD

STOP - To halt the object program temporarily or permanently.

STORAGE - See glossary

SUBTRACT - To subtract one quantity from another and store the result in the last-named field or a specified field.

SWITCHES

TAPE(S) - Magnetic tapes.

THAN

THREE(S) - A figurative constant used in procedure sentences.

TO

TOP

TRUE~FALSE - An optional Data Division section name.

TWO(S) - A figurative constant used in procedure sentences.

TYPEWRITER - Control Console Typewriter.

UNEQUAL - Not equal to.

UNTIL

USE

USING

VARY - To start and control repeated execution of a sentence.

WITH

WORDS - A basic unit of information in the GE-225.

WORKING (~STORAGE) - A mandatory Data Division section name.

WRITE - To display a limited amount of information on the console typewriter.

-To release a record or group to an output file.

ZERO(S) - A figurative constant used in procedure sentences.

ZEROES - SAME as ZERO(S)

APPENDIX 2. SUMMARY GUIDE FOR GECOM FORM PREPARATION

The following pages briefly summarize the basic rules to be followed in preparing GECOM source programs on the General Compiler Sentence and Data Division Forms. A copy of this appendix is used to provide novice programmers with a convenient guide and a ready reference while becoming familiar with GECOM.

SUMMARY GUIDE FOR SENTENCE FORM PREPARATION

A sequence number can be assigned in columns 1-6 for each line (card).

All division, section, and sentence names should be started in column 8 (recommended). These names must be followed by a period and at least one space. Sentence names are limited to 12 characters.

Each sentence must start on a new line. However, the first sentence following a sentence name can start on the same line as the name.

Although unnamed sentences can start in column 8, it is recommended that they be indented and start in column 12.

If a sentence exceeds one line, it can be continued on the next line starting in column 8. However, it is recommended that continuations be indented 8 spaces and started in column 16.

~ If a word is split at the end of a line, place a tilde in column 7 of the 2nd line and continue the word. If splits are avoided, space-fill the 1st line thru column 80 and continue on second line at column 8 or 16

SUMMARY GUIDE FOR DATA DIVISION FORM PREPARATION

1 11

DATA DIVISION. Starts in column 8, ends with period. No other entries.

ARRAY SECTION.
TRUE~FALSE SECTION.
INTEGER SECTION.
} Optional sections as required by program. Start in column 8 and end with a period.

FILE SECTION. Identifies characteristics of data in input and output files of the object program. Starts in column 8 and ends with a period. Mandatory section.

OUTPUT FILES. Introduces output file descriptions. Starts in column 8 and ends with period.

INPUT FILES. Introduces input file descriptions. Starts in column 8 and ends with a period.

WORKING~STORAGE SECTION. Introduces working storage descriptions. Starts in column 8 and ends with a period. Mandatory.

COMMON~STORAGE SECTION.
CONSTANT SECTION.
} Optional sections as required by program. Start in column 8 and end with period.

FD File description. Name follows in columns 11 through 22, 12 characters or less.

2 11

INPUT RECORD ENTRIES:

R Input record-Name in column 11 through 22, 12 characters or less.

P Indicates all fields in the record are
U decimal and packed (P) or unpacked (U).
If P, then no entry can be made in column 43, Binary.
, Indicates all fields in the record are separated by commas.

*G A group of fields which may be referenced by WRITE and made available by READ. Must be followed by another *group or be last record-entry. Name is entered in columns 11 through 22.

P Indicates all fields in *group are
U packed or unpacked.

G A group of fields within a record; cannot be referenced like a *group. Name is entered in columns 11 through 22.

P Same as *group above.
U

3 11

F Indicates a field of an input record.
Field name is entered in columns 11 through 22.

P Assumes field is packed or unpacked,
U unless it conflicts with a higher level entry (group, record, or file).

1 Assumes one-word binary numeric data. If the data is not integer, a scaling factor must be supplied in the data image columns.

2 Assumes two-word non-standard binary numeric data. If data is not integer, see note above.

S The preceding image is to be used for this entry. Cannot be used if preceding image has a 1 or 2 in column 37.

If any input groups or fields are repeated consecutively, the number of times repeated is entered here.

4 11

E Element name in columns 11 through 22.

Most significant position of element in the field.

Least significant position of element in the field.

Elements require no further data Their descriptions follow the descriptions of their parent fields.

B or other character assumes that levels with numeric descriptions are to be in GECOM binary form unless in conflict with a lower level format entry in column 37. If column 37 contains P at a given level, column 43 must be blank for that level. A blank in column 43 assumes BCD data. Entries at lower level take precedence.

5 11

FL Field literal. Any legal data name. Used for named fields with fixed values. Rules that apply to fields also apply to field literals. Actual value of literal is enclosed in quotation marks in columns 55 through 80.

OUTPUT RECORD ENTRIES:

R Output record-Name in columns 11 through 22; may be qualified by entry of a qualifier in columns 24 through 35. If record name is unique, It need not be qualified.

P Forces all levels within record to be
U packed (P) or unpacked (U) except binary numerics.

*G *group name in columns 11 through 22. May be qualified. If 2 qualifiers are needed, first goes in columns 24 through 35, second in
~ next line columns 24 through 35 and a tilde in column 7.

P Forces lower levels to be
U packed or unpacked.

SUMMARY GUIDE FOR DATA DIVISION FORM PREPARATION (continued)

6 11

G Group name in columns 11 through 22. May be qualified as is *group.

P Forces lower levels to be packed
U or unpacked. High level nonblanks take precedence.

F Output field name in columns 11 through 22. May be qualified as is *group.

P Forces field to be packed or unpacked
U unless higher level entry conflicts.
1 Assumes one-word binary data. If not integer, scaling factor must be in data image.
2 Assumes non-standard 2-word binary data. If not integer, supply scaling factor.
S Repeat preceding image. Cannot be used if preceding image is a 1 or 2.

E Element entry and name for documentation only. No other entries required. These entries are ignored during compilation. If element is required for processing, rename as a field.

7 11

L Literal; no name used. All other columns are completed as for fields and elements.

OTHER OUTPUT RECORD ENTRIES

Not used for output entries.

B or other character forces lower levels with numeric data description (9) to be in standard binary form unless lower level Format indicates non-standard binary data. A blank in column 43 forces BCD data output.

Forces unpacked data to be left L (L) justified and zero filled or R right (R) justified and blank filled.

SUMMARY GUIDE FOR DATA DIVISION FORM PREPARATION (continued)

8 11

DATA IMAGE ENTRIES:

INPUT AND OUTPUT ENTRIES:

Position contains one of the characters of the GE-225 character set, alphabetic or numeric.	X
Position contains a leading or trailing blank; not a significant character.	B
Position contains a leading or trailing zero and is not a significant character.	0
Position contains a plus sign when field is positive and a minus sign when field is negative.	+
Position contains a minus sign when field is negative and is blank when field is positive.	

SUMMARY GUIDE FOR DATA DIVISION FORM PREPARATION (continued)

		9 11
Position contains an alphabetic character, A-Z, or a blank.	A	
Position contains an integer 0-9.	9	
Position contains a numeral 0-9 with an 11-row overpunch when negative and no overpunch when positive.	R	
Position contains a numeral 0-9 with a 12-row overpunch when the field is positive and an 11-row overpunch when the field is negative.	I	
Indicates an assumed decimal point. Neither the V or the decimal point occupy an actual field position.	V	
Indicates number following E is a power of ten to which the number preceding the E must be raised. E does not occupy field position.	E	

	10	11
Scaling factor for binary data. Follows decimal description of field, indicates binary point. For non-standard data.	S	
INPUT ENTRIES ONLY:		
Position contains a minus sign when field is negative and most significant figure when field is positive. may be used only when fields are separated by commas.	T	
Indicates that zeros must be applied to the recorded value of field to obtain true value. Can indicate left or right scaling.	P	
OUTPUT ENTRIES ONLY:		
Inserts dollar sign in output. Must be in 1st position of field.	$	

		11	11
Inserts a comma in corresponding field positions. Automatically suppressed by floating dollar signs, zero suppression, asterisk filling.	,		
If position occupied by Z in numeric field becomes zero, zero is suppressed and position prints blank.	Z		
If position occupied by * becomes zero, * is printed.	*		
If position occupied by $ in numeric field becomes zero, move $ into it.	$	$	

END PROGRAM. The final entry of the data division must be END PROGRAM starting in column 8 and terminating with a period.

APPENDIX 3. SOURCE PROGRAM ORDER FOR COMPILATION

I.	IDENTIFICATION DIVISION	Mandatory
	PROGRAM~ID.	Mandatory
	NEXT~PROGRAM	Optional
	AUTHOR.	Optional
	DATE~COMPILED.	Optional
	INSTALLATION.	Optional
	SECURITY.	Optional
	REMARKS.	Optional
II.	ENVIRONMENT DIVISION.	Mandatory (whether or not any sentences follow)
	OBJECT~COMPUTER.	Optional
	I~O~CONTROL.	Optional
	FILE~CONTROL.	Optional
	COMPUTATION~MODE.	Optional
III.	PROCEDURE DIVISION.	Mandatory
	Closed sections and decision tables delimited by BEGIN-END	Placement mandatory if sections are used.
	Master program	Mandatory
IV.	DATA DIVISION.	Mandatory
	ARRAY SECTION.	Optional
	TRUE~FALSE SECTION.	Optional
	INTEGER SECTION.	Optional
	FILE SECTION.	Mandatory*
	OUTPUT FILES.	Mandatory*
	INPUT FILES.	Mandatory*
	WORKING~STORAGE SECTION.	Mandatory*
	COMMON~STORAGE SECTION.	Optional
	CONSTANT SECTION.	Optional
	END PROGRAM.	Mandatory*

* The section heading card is mandatory; further entries under it are optional.

APPENDIX 4. GLOSSARY

A list of important terms (most of which are used frequently in the body of this manual and many of which are encountered frequently in other GECOM literature) have been included in this glossary. Most definitions are deliberately brief and are not intended to be comprehensive; many of the terms have additional meanings. For more detailed and more exhaustive listings, the reader is referred to any of several excellent glossaries of information processing terminology.

ADDRESS - A specific location in storage or memory. Actual addresses are numeric. Addresses used in GECOM are symbolic, that is, represented by names.

ARITHMETIC EXPRESSION - A sequence of data names, numeric literals, and/or mathematical functions connected by mathematical symbols.

BCD - Binary Coded Decimal; a system for representing any character of the character set of the computer by a group of binary digits.

BEGINNING FILE LABEL - A group of records (blocks) which identifies a file in a multifile magnetic tape. It is block 0, the first block of each file.

BINARY NUMERIC - A digit or group of characters or symbols representing the total units using the base two; a number expressed in binary digits or bits, 0 and 1.

BLOCK - A group of records read from or written on magnetic tape as a single physical tape record.

BLOCK SIZE - The number of words in a block.

BUFFER - Storage locations used to compensate for differences in rate of data flow when transmitting data from one device to another.

CHARACTER - One of a set of basic symbols used to express data. Includes decimal digits 0 through 9, the letters A through Z, punctuation, and special symbols.

CONDITIONAL EXPRESSION - An expression that can be either true or false.

CONDITIONAL NAME - A name assigned to a possible value of a numeric or alphanumeric field or element. A conditional name must be described in the Data Division.

CONSTANT - A value used in a program without alteration. Constants are either literal, figurative, or numeric in GECOM.

DATA IMAGE - The characteristics of a data field; that is, length, content, sign, and character type for each position. The data image is used within the Data Division to define data input and output.

DATA NAME - A programmer-assigned word naming a file, record, field, constant, or other data. Data names are composed of letters, numerals, and hyphens, not exceeding 12 characters, and may be names of records, groups, fields, arrays, elements, sections, or true-false variables.

ELEMENT - A subdivision of a field. For example, a date field could contain a DAY element, a MONTH element and a YEAR element.

FIELD - A unit of data within a record. It may or may not be a part of a group.

FIGURATIVE CONSTANT - A special name representing specific values [ZERO(S), ZEROES, SPACES, ONE(S), through NINE(S)]. May be used in procedure sentences to imply strings of characters.

FILE - A set of records

FIXED-POINT - A number which includes a decimal point, either between digits or following them (1.23, 123., or 123.0)

FLOATING-POINT - A number expressed as a whole number, a decimal fraction, and a power of ten. ($1.287*10^{-2}$)

GENERATED FIELD - A field (of data) which is generated as a result of calculations and is not input to the program.

INSTRUCTION - A group of symbols causing the data processor to perform some operation.

INTEGER (as used in this manual) - A number of 5 digits or less not containing a decimal point.

LEVEL - In GECOM, the relative status of one item of information to another. For example, records are on equal levels, fields are on lower levels than records.

LITERAL CONSTANT - A specific unit of data that remains unchanged throughout the program. It may consist of up to 83 GE-225 characters in any combination and is enclosed in quotation marks to indicate that it is literal and may not be used in arithmetic operations.

LOGICAL EXPRESSIONS - A combination of conditional names, relational expressions, and arithmetic expressions connected by the logical AND, OR, and NOT.

LOGICAL RECORD - Any consecutive set of related data within a physical record.

LOOP - A coding technique permitting repetition of a group of instructions under program control.

MACHINE LANGUAGE - The coding system for representing instructions and information within an information processing system.

MEMORY - Main storage, core memory, core storage.

MULTIFILE TAPE - A magnetic tape containing more than one data file.

MULTIREEL FILE - A data file exceeding the capacity of one magnetic tape reel.

MULTITAPE FILE - Same as multireel file.

NONSTANDARD DATA - Data not conforming to GECOM internal binary scaling.

NUMERIC CONSTANT - A specific unit of data that remains unchanged throughout a program. It may be an integer of up to 5 digits, a fixed-point number of up to 11 digits (not counting sign and decimal point), or a floating-point notation with an exponent between ±75 and a mantissa of 9 or less digits (only one of which is to the left of the decimal).

OPERAND - In GECOM, the object of a verb; the receiver of action, or that which is operated upon. In GE-225 machine language, the address portion of an instruction.

OBJECT PROGRAM - A program in machine language. An output of the General Compiler.

PHYSICAL TAPE RECORD - Information contained between successive magnetic tape gaps (unrecorded areas).

PROCEDURE NAME - A programmer-assigned word naming a sentence or a section of a sentence. Procedure names are composed of letters, numerals, and hyphens, not exceeding 12 characters.

QUALIFIER - A data name used with nonunique data names to make them unique.

RECORD - An accumulation of related data, usually in fields, input or output from a data processor.

RECORD SIZE - The number of words in a record.

RELATIONAL EXPRESSIONS - An expressed or implied comparison of two field or element names, literals, or arithmetic expressions.

SENTENCE - A GECOM procedure statement.

SOURCE LANGUAGE - A vocabulary having special meaning for the General Compiler.

SOURCE PROGRAM - The English-language program written for the General Compiler.

STATEMENT - In GECOM, a collection of words, usually including a verb, specifying one or more operations.

STORAGE - See MEMORY.

STORED PROGRAM - A data processing program contained within the information processing system and occupying storage as does the data to be processed. The program thus can be manipulated as if it were data.

SUBROUTINE - A series of instructions to perform an operation; a procedure.

SUBSCRIPT - A method for identifying or selecting a particular value in an array of values.

TAPE LABEL - A 24-word binary record that identifies a tape, or a file, within a multifile tape. GECOM permits four types of tape labels: beginning-tape (BTL), beginning-file (BFL), end-tape (END REEL), and end-of-file (END FILE).

TAPE MARK - A special character (001111) that signifies either end-of-file or end-of-tape, depending upon the fence block following it.

TRUNCATION - The dropping of the least significant digits of a number without rounding off.

VARIABLE - In GECOM, a field in storage that may assume different values during object program processing.

VERB - In GECOM, a word that causes a data processing operation to occur.

WORD - (in data processing) - A set of characters which is moved as a unit by the computer. A word may be data or instruction.

Progress Is Our Most Important Product

GENERAL ELECTRIC

COMPUTER DEPARTMENT • PHOENIX, ARIZONA

CPB 230

LITHO IN USA