



LAMBDA

LAMBDA DATA MANAGEMENT SYSTEM

December 1970

LAMBDA CORPORATION  
1501 Wilson Boulevard  
Arlington, Virginia 22209

## INTRODUCTION

In recent years managers in industry have become increasingly aware of the value of timely, relevant information about the operations under their control. Part of this awareness is due to the complexity of today's operation, compared with only a few years ago; and this is as true of unified firms as it is of multi-divisional and conglomerate operations. Some of the awareness undoubtedly also springs from the fact that managers are expected to do more, and therefore to know more, than managers of a decade or two ago.

The natural outgrowth of this need for information is an almost universal drive to enhance and expand the system for delivering management information. Yet an organization that is considering improving its system is faced with a bewildering array of decisions. Should it develop a totally new system completely junking the old? Can it improve the existing system with relatively minor changes? Can software packages meet our requirements?

There are, of course, basic requirements for any management information system: It must be able to gather and retrieve the traditional data required to operate the firm, including inventory control, personnel records and payroll, production monitoring and so on. We have found that most organizations are able to design and install a system that provides these basic kinds of data. Most of these systems suffer, however, from one or more of the following flaws.

- Sometimes the basic system consists of independent subsystems -- one for personnel and payroll, another for sales data, and so on. When a manager needs to blend together information that is systematically separated, he finds that it can be done only with a great "manual" effort.
- Whenever an organization must adapt to new conditions, or when new and critical problems arise, it is found that the nature and design of the information system makes the change exceedingly difficult -- or costly. This can be traced in many instances to the format in which the files are kept and to the interrelatedness of the software components of the system.

- Typically, when a manager examines his standard operating and financial reports, he asks questions that cannot be answered before decisions can be made wisely. Sometimes, of course, such a question will be recognized as one that will frequently arise, and a decision will be made to include it in the list of standard reports. It is a mistake, however, to enlarge the standard reports each time a special need arises, since this leads to an unwieldy and unusable set of reports.

Lambda Corporation has developed a family of techniques and programs for data management that surmount these difficulties. They

- provide all the basic operating, financial, and personnel reports
- make it possible to gather and transfer information across the whole range of an organization's files
- simplify the process of modifying the information system
- incorporate as a basic part of the system the ability to respond quickly to "special" requests.

## THE LAMBDA SERVICE

We do not offer a "data management software package." We have found that, regardless of the many similarities between two organizations, the differences between them control the design of their information systems. The cost of tailoring our programs to each user's needs is considerably less than the cost, in dollars and limited capability, of forcing those needs to be satisfied by the constraints of a fixed package. This is confirmed by the general disappointment encountered by most users of data management packages.

Our procedure is generally as follows.

- Survey and plan. In a one- to two-week survey phase we review the existing system, its forms and procedures, and develop a detailed plan including cost and schedule projections.
- Review of data and data sources. We review procedures used to gather and prepare data and attempt to identify sources of inconsistencies. Using a specially designed program, we convert files to a new integrated data base. Together with the client's staff, we suggest improved procedures.
- Design the system. We select components that are suitable to the organization's overall design and create any new components that may be required.
- Implement the initial system. In general, we recommend that the initial system not stress sophistication. It is virtually impossible to predict accurately, in the design phase, the final system configuration that will work best in an organization; such guidance is best obtained from actual use.
- Modify and enlarge the system. As experience grows, changes can be made -- and, because of the built-in flexibility of the Lambda concept -- can be made easily.
- Train users and EDP personnel. We accept full responsibility for designing and installing the system, including training. Part of this effort is the development of new documentation tailored to the needs of the organization, not merely a restatement of the basic software programs provided by Lambda.

## THE LAMBDA DATA MANAGEMENT SYSTEM

Lambda's data management software system is particularly well suited to any organization that has a large and complex data bank and that desires great flexibility in the reports it may need to generate. This flexibility is essential because it is very difficult for organizations to specify the exact routine reports, management reports, and file maintenance capabilities desired of their software before systems are in operation. This is especially true if there is going to be any attempt to use information contained in computer files to prepare analyses and reports needed by top management.

Our system can:

1. Read all the existing files;
2. Integrate these files as required for reports;
3. Prepare routine reports (payroll, personnel, appropriations, expenditures);
4. Accept modifications and changes to the integrated files; and
5. On the basis of simple English language instructions, prepare a broad range of management reports.

As a result of facing similar requirements before, we have developed what we believe is a set of data management computer programs and subroutines that are unique in two ways. First, they can be adapted for use in virtually any particular application. This has the dual effect of reducing the cost of preparing software for a new application and tremendously increasing the reliability of the resulting software.

Second, our programs are based on radically different data management programming concepts, which allow the tremendous flexibility inherent in our systems. One can add new types of information to a file, specify changes to the file in new ways, and prepare new aggregations, summaries and reports without writing any new programs. At the same time, the software allows performance of utility functions such as payroll, budget, and financial reports in at least as efficient a manner as conventional systems. (In most applications, the efficiency of conventional functions is actually improved.) But again, the key point is that using new design concepts, we have prepared software which allows an information system to be built in an evolutionary way. It is virtually impossible to "paint yourself into a corner". If something is left out or the requirements change, the system can be adapted with virtually no reprogramming.

What is perhaps the key element of our data management concept -- the "attribute-value" concept -- can be described quite simply. We carry the specification of the logical meaning of a data element along with the element, while in a conventional system, data are defined by their position in a file. To add a number (such as 10,000) representing an individual's annual salary to an attribute/value file, one would add a pair of items -- the specifications of the "attribute" (SALARY), plus its "value" (10,000). In a conventional system, only the 10,000 would be added to the file. The meaning of the number would not be specified explicitly but would be defined by its location within a "record". For example, the conventional file might be

defined as a series of records of 100 characters each with the first eight characters containing an individual's salary, the next twenty his name, etc.

An attribute/value file would contain a succession of pairs, such as SALARY, 10000; NAME, JOHN DOE; AGE, 45; etc., with each logical grouping of such pairs set off by a delineator such as "END OF ITEM." We call such logical groupings "items;" as can be seen, an item is logically analogous to a record.

The trouble with records is that the formula have to be defined in advance, and their definition differs for every application.\* If once the record definition changes, all the programs written to read, manipulate, extract and sort the records no longer work. It is impossible to write general-purpose software to handle records defined for specific purposes. Thus, only by breaking away from the use of records altogether, a radical departure from conventional procedures, can one develop software with the flexibility truly needed in most applications.

In summary, the attribute/value system has the following important advantages over a conventional formatted-record system:

- Increased reliability through the use of proven, "debugged," components. Systems using the same subroutines are in operation in a number of installations.
- Tremendous flexibility, allowing evolutionary development of information systems.
- No limitations imposed by record size or format.

---

\* In addition, an attribute/value system overcomes some computational inefficiencies associated with using files composed of records which can be overcome. These are briefly described in Appendix 2, which also describes in further detail the difference between attribute/value and conventional files.

- General purpose report generator for all tabular reports.  
Appendix 1 gives a comparison of conventional procedures required to request a new report and the procedures that would be required using the Lambda system.
- Ability to accept initially all information in all existing files, allowing them to be integrated as inconsistencies are corrected.
- More efficient file storage. As explained below, the nature of the Lambda system usually allows quite dramatic compression of data files.
- The entire system can be readily adapted to different machines.
- Modularity. A complete separation of components with regard to logical function into separate programs and subroutines allows improvements and modifications of any component at any time without interference with the rest of the system.

We should emphasize that we do not offer for sale a "software package".

While our programs and subroutines have been developed and used elsewhere, there is always a considerable amount of adaptation needed to satisfy the particular machine hardware characteristics and operational desires of an individual user.

We have found that the cost of tailoring our programs to each user's needs is considerably less than the cost of trying to make a simple data management system or "package" which will run on any machine for any user. In fact, we believe that the latter is probably impossible except for extremely well-defined tasks. This is evidenced by the general disappointment encountered by most users of data management software packages.

## Procedures

In general, we form a task force jointly with the client's personnel. This task force would consist of at least one member of each of the functional areas affected by the Data Management System (Budget, EDP, Payroll, Revenues). Each member would be responsible for assisting in the design and implementation of the system in areas of his functional specialty. Usually the major subtasks to be accomplished by the task force are:

- The explicit definition of the data currently collected and maintained in the payroll and revenue files;
- The construction of a "directory" (defined in Appendix 2) containing definitions of all attributes and values;
- The conversion of the present files to an Attribute/Value Data Base using Lambda software;
- The reconciliation of inconsistencies in the files;
- The exact specification of procedures used to maintain existing files; and
- The development of improved procedures to gather and prepare data for insertion into the data base.

The sub-tasks usually accomplished by Lambda are:

- The adaptation and installation of all additional software;
- The preparation of program input cards needed for conversion of existing tapes; and
- The testing of the entire system.

The degree of success of the project depends heavily upon the cooperation of the members of the client's staff. Their expertise and knowledge of their functional areas is a limited and valuable resource. In recognition of this fact, Lambda makes every effort to incorporate their ideas and suggestions into the design of the system.

The data-management software delivered and installed on the client's computer would contain the following major components:

- A data file in attribute value form and a general-purpose translation program, called CONVERT, which will allow translation of not only existing files but virtually any data files from conventional to attribute/value form.
- A report generator program, called AVREGE, which allows sets of data in the data base to be extracted, and then combined, broken down or manipulated in almost any combination without further programming, and hence produces virtually any type of tabular reports for which the file contains the needed data.
- A report generator program, called RG-4, which allows extraction of data in such a way that it can then be used by any FORTRAN subroutine. [AVREGE and RG-4 between them allow the full range of data processing and calculational (or "scientific") operations to be performed very simply on the entire data base in one pass through the file without intermediate input/output operations and complicated report requests.]
- Report generators for utility and routine reports (budget detail listing, payroll listing, etc.).
- File maintenance programs that will allow the integrated file to be changed or updated simply by specifying the set of items to be changed in a simple English language form (change all clerical wages in Department A) and the change to be made ("multiply wages by 1.1").

- PROGEN, a program to record all modifications to any part of the data management system, which by automatically providing a record of all changes to the system allows reconstruction of past programs if these are needed, the reversal of any changes; PROGEN also facilitates up-to-date documentation of the system throughout its lifetime, and comparison of old and new reports on a comparable basis.
- User's documentation which will give client personnel complete instructions on how to use all programs and systems delivered.

**APPENDIX 1**

**A COMPARISON**

Figure 1 diagrammatically represents the report request flow of a typical conventional system. The process begins when the user requests EDP to prepare a report. This report request must specifically describe the location and form of the data as well as the desired output. EDP then determines if the request can be met without compromising other responsibilities. The decision depends on the availability of programming in the center and is a critical factor in the process. If it is determined that the report can be processed, assignments are made and the project enters the job queue. Upon exiting the queue, the actual development is begun. In the case of a report that had never been requested previously, a set of programs is designed to extract the data from the data base and manipulate it into the form required. This task is rarely straightforward and usually requires a great deal of time and effort. If, on the other hand, the reports are standard (utility reports produced at scheduled times), the task is reduced to checking existing programs and processing the request. This is frequently complicated by the fact that changes have occurred in either the processing system, the data base formats or the file maintenance system since the last application of the programs (the nature of field position definition of data necessitates format changes whenever addition or deletion of characters occur). In either case, the report is processed once the tested and debugged software is complete.

Figure 2 outlines the request flow of the Lambda system. In this process, the user communicates his request to a designated individual in EDP in English language terms. The contact transcribes the necessary information into report request form and has it keypunched. The punched cards are then added to a job deck containing other requests received that day and processed when computer time is available. In the case of an emergency request the processing takes place immediately and in either case the reports are delivered as soon as possible after processing. A specific example will be of aid.

If the user required a breakdown of hospital costs by program for Fiscal Year 1971, with program broken down into cost groups and cost centers, then

Step 1 - The user calls EDP and communicates the above request.

Step 2 - The contact fills in the request form with the following information:

Card 1	Quantity	Hospital	(Blank)	Row
Card 2	Total	Program	(Blank)	Cost Group
Card 3	(Blank)	Cost Center	(Blank)	Column
Card 4	Fiscal Year	List	FY 71	End
Card 5		End		

Step 3 - The cards are punched and submitted for processing.

Step 4 - The job is processed.

Step 5 - The report is produced.

One page of the report requested above is represented in Figure 3.

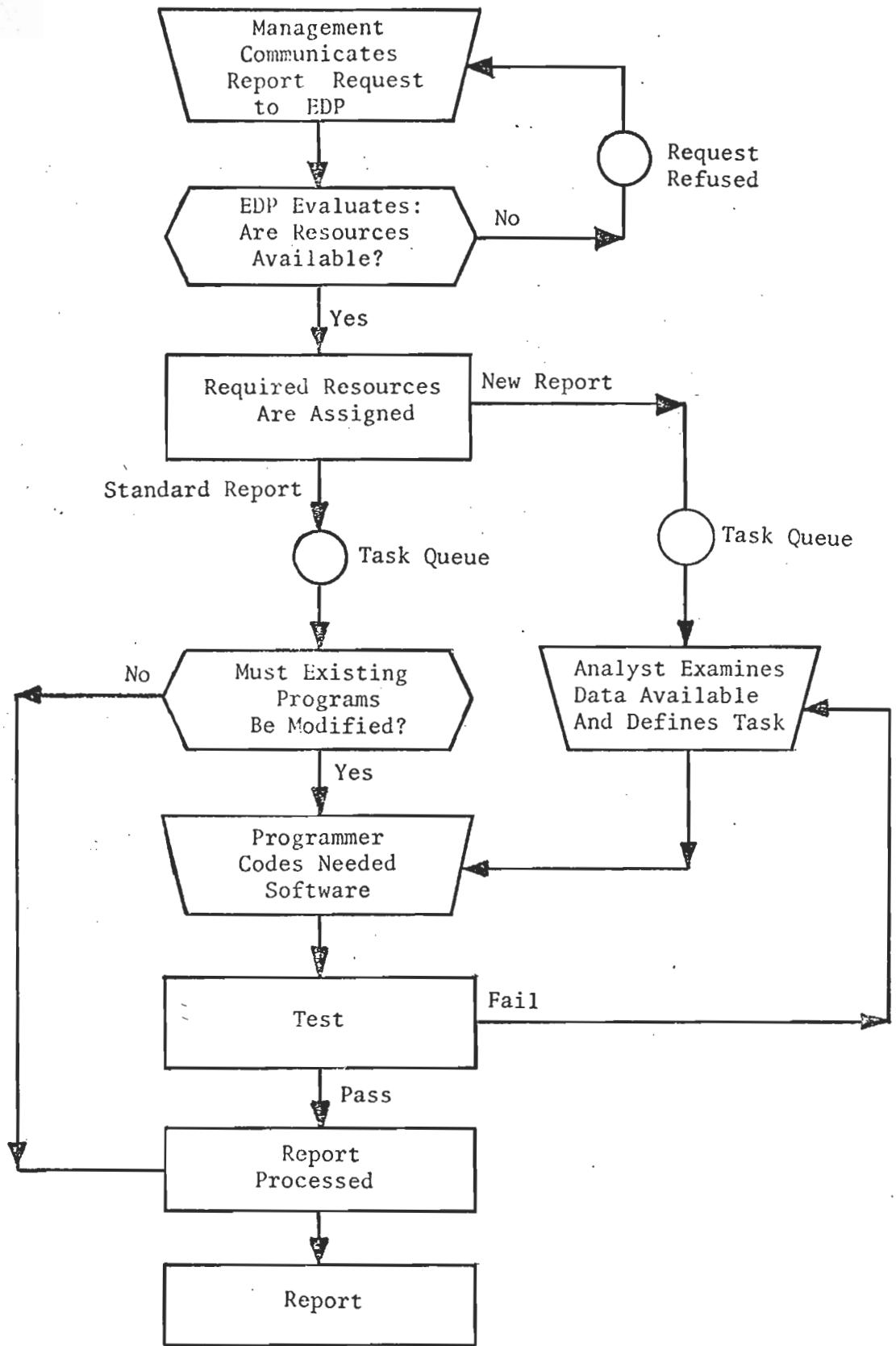


Figure 1. Flow of Report Request in Typical System

### Major Negative Features

- \* The entire process can take as long as a month and rarely takes less than a week.
- \* The scarce resources of EDP are tied up for extended periods of time on what should be a routine process.
- \* Many badly needed reports are never processed or are received too late to be of use (due to the limited number of available programmer/analyst and not to a lack of cooperation on the part of EDP).
- \* The user must spend a great deal of his time working with EDP on specifications.
- \* Existing programs are continually being made unworkable by changes in the process or data base.
- \* Potential users abstain from requesting needed information due to the above features.
- \* The limit of the system is a function of the availability of manpower in EDP and not the need of the user.

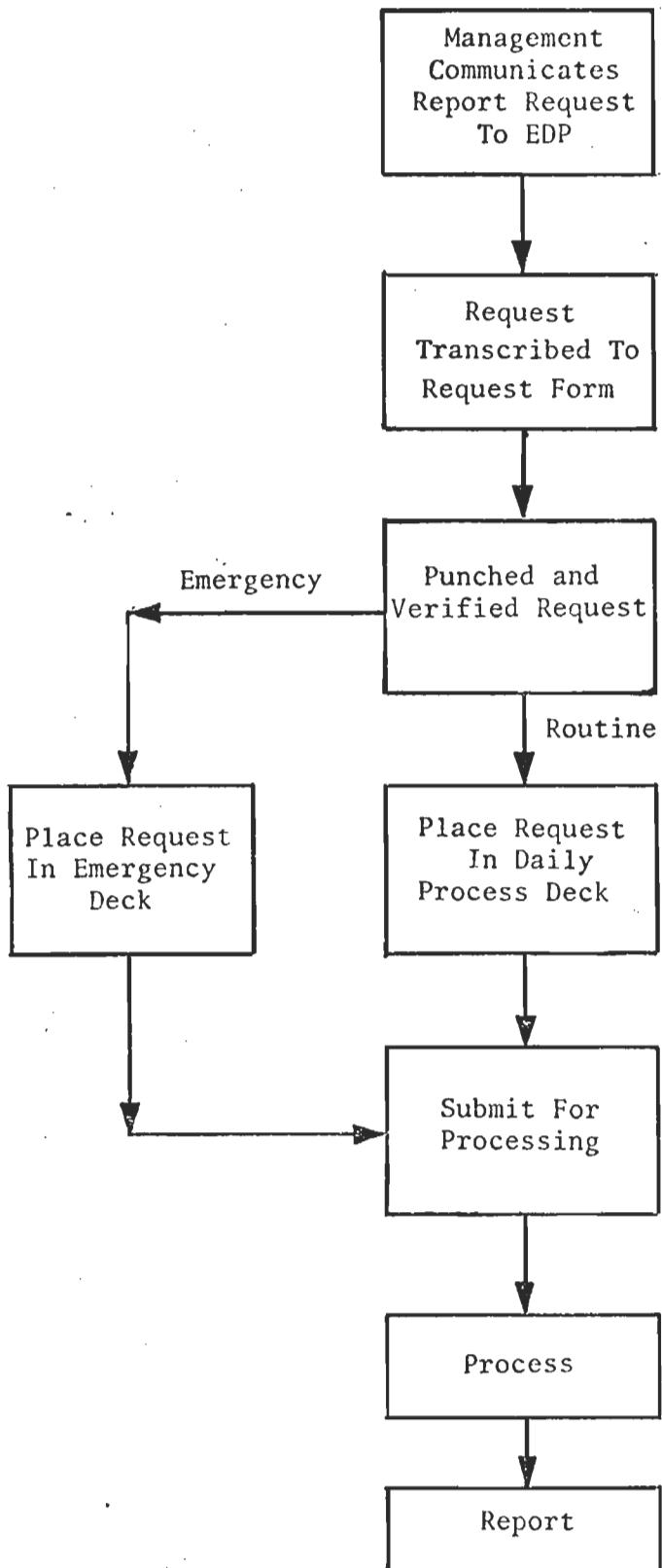
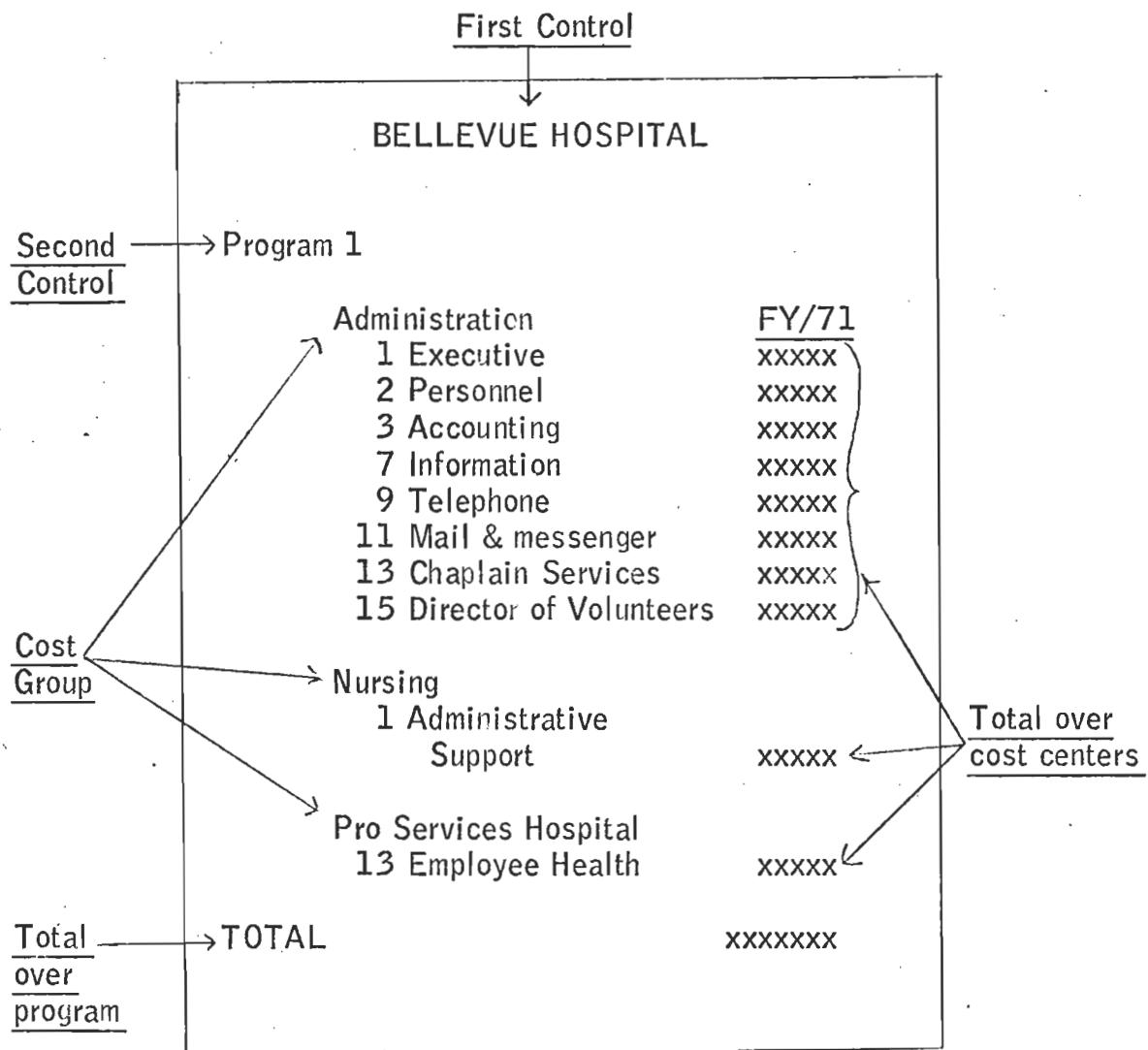


Figure 2. Lambda Report Request Flow

### Major Advantages of the Lambda System

- \* The process from request to report takes only a fraction of the time presently consumed (usually 25 hours).
- \* The limit resources of EDP are free to be used on more important projects.
- \* Information is provided when it is needed and not when it can be made available. Requested reports are received on virtually a real-time basis.
- \* The user need not become involved in technicalities of EDP. He simply requests his report in English language terms.
- \* Changes in one component of the system have no effect on the operation of the system.
- \* The system is capable of supplying information to anyone who has a need.
- \* The limit of the system is a function of the needs of the user and not the manpower available in EDP.

## EXAMPLE OF POSSIBLE REPORT



The user request which would generate this type of report is for a breakdown of Bellevue's budget for Program 1 (HOSPITAL ADMINISTRATIVE SUPPORT) by Cost Group and Cost Center within Group, with totals over Cost Center and Program.

Figure 3

## **Appendix 2**

### **The Attribute-Value System**

Attribute/Value (A/V) data files are mentioned in the general description of the system. However, several additional technical points should be made.

1. Although both attributes and values are kept in the file, whereas only values are kept in a conventional file, A/V files are usually smaller than conventional files. This is because A/V form allows keeping the file in "differential" form. For the first logical grouping of A/V pairs (item), a value is recorded on the tape or disc for each attribute. Thereafter, values are recorded only if they differ from the value in the previous item. They, if sequential "records" (items) were all for the same hospital, for example, the value of the attribute "hospital" would be recorded only for the first of these items.

2. Another factor tending to reduce the size of files is the use of a "directory." For attributes which can have only a reasonably small number of possible values (usually less than 400), the values are not stored directly in the file. Rather, they are assigned an index number according to their location in a directory and the index number is stored. Attributes are also assigned index numbers. For example, if "Hospital" were assigned attribute index number 8 and "Bronx Municipal" were assigned value index number 5, the tape or disc would contain only "8, 5" to denote "Hospital, Bronx Municipal." This obviously saves considerable space.

3. The index numbers are automatically assigned by the system. They are not "codes" assigned by the users. The user simply specifies the attributes and values to be included in the directory.

4. If an attribute can have so many possible values that it would be cumbersome to enumerate them, the attribute is said to be "type dynamic." \* In this case, the values are stored directly in the file.

5. All report requests, output reports, and file maintenance is done by the user using actual attribute names and values, not index numbers. The directory is stored internally. Names are converted to index numbers and vice versa automatically.

---

\* The name "dynamic" comes from the fact that one Lambda report generator program, AVREGE, "dynamically" builds a directory for these attributes.

6. Using a directory has other advantages in addition to allowing compression of an existing file. First, the directory provides an exact description of the contents of the file at any time. This allows new users to easily determine if the information they seek is in the file. Second, through the use of the directory, English language descriptions can be used for items that would normally be too long to include directly in a file. Users do not have to memorize or look up the meaning of large numbers of "codes."

7. In addition to its flexibility and ease of use, our report generator AVREGE has significant efficiency advantages over conventional report generators. First, many report requests can be processed with a single reading of the file. Second, only the information needed for each report is sorted, not the entire file.

The attached charts give an illustration of a conventional record, the way it would be included in an attribute/value file, and a directory. Note that nothing analogous to the directory is included in the conventional system. The definition of record fields is only implicit in a conventional system.