A STUDY OF THE GENERALIZED PROGRAMMING CONCEPT FOR A DATA RETRIEVAL APPLICATION

A Thesis

Presented to

The Faculty of Graduate Studies and Research

The University of Manitoba

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Computing Science

---

by

Richard John Keltie, B.Sc. (M.E.)

May, 1967

ABSTRACT

This thesis studies the concept of generalized computer programming as it relates to data retrieval. Of particular interest here is the problem of data retrieval from free formatted files in which data is stored in random sequence.

The investigation is divided into two parts. In part one a study is made of the state of the art of generalized data retrieval programs, while in part two a problem is solved for data retrieval from free formatted files using the concept of generalized programming. The programs were written for the IBM/360 and involved a combination of the Report Program Generator language and Basic Assembler Subroutines.

This approach proved successful for small to medium size files and involved a minimum of programming effort. For large files a more comprehensive study is suggested although not investigated. Generalized programs are shown to involve quite advanced programming effort. Although the concept is well known, it is still not very common nor is it used in the case of complex retrieval requirements. To date it is still very much in the development stages. In summary this thesis offers further refinement to the development of generalized data retrieval computer programming.

TABLE OF CONTENTS

List of Figures

ACKNOWLEDGEMENTS

The author is indebted to all those who have contributed or assisted in making this thesis possible.

A special thanks is due to Professor B. A. Hodson, Director of the University of Manitoba Computer Centre and Eugene Lashyn and Irene Pran of the International Business Machines Corporation, Winnipeg, whose technical advice and encouragement throughout the investigation was appreciated.

The Canadian Wheat Board and Air Canada in Winnipeg, MacMillan Bloedel in Vancouver, and the University of Manitoba Computer Centre who supplied program test time free of charge, and the National Research Council whose financial help made this study possible are sincerely thanked.

PART I

A DESCRIPTION OF GENERALIZED DATA RETRIEVAL

CHAPTER I

INTRODUCTION

With the recent advent of the "third generation" computers business data processing is moving into a new area of computing. It is computing in which up to the minute status of a company's file of information can be queried at any time by the management. This concept is termed a real time company-wide information system. To study some of the problems involved in developing these information systems is the overall purpose of this thesis.

It is actually three studies in one paper. Firstly, it is a study of the concept of Generalized Data Retrieval. In particular a description is given of the IBM Report Program Generator for the system /360. This is a generalized data retrieval program which is now finding wide use in the computer industry. Secondly, this thesis is a study of a unique type of file organization for information storage. The file is organized on the basis of time, containing information which is stored as it is received from the environment, in time sequence. Finally explanation is given of an attempt made to use the generalized data retrieval program for the /360 to retrieve data from the time sequence file.

But how is generalized data retrieval and the time sequence file organization related to company-wide information systems? These information systems are very complex and costly to develop. They involve many man years of human effort in interviewing, system designing, and program writing and testing. Any techniques which can facilitate the development

of these information systems are of great interest to the computer oriented industries. Generalized data retrieval and the time sequence file are two suggested techniques.

A second article from the Financial Post may illustrate the above point in another light.

> New aids to business are arriving on the market at an unprecedented rate of technological innovation. These aids will greatly assist businessmen to meet growing world competition and in solving problems of cost, profit and employment (20).

Third generation computers are the aids he is suggesting. They have been designed to operate closely with businessmen in decision making processes. Inside the computer field aids are being developed to assist the computer to assist the businessman in making his decisions. So these computer aids directly touch the effectiveness of businessmen to compete. Two such computer aids suggested in this paper are generalized data retrieval and the time sequence file organization. The rest of this paper is an attempt to prove their utility in computerized industry today.

CHAPTER II

THE GENERAL EXPLANATION

Generalized data retrieval is a concept in programming used for the retrieval of information from files. This chapter will outline its approach under the following three headings, concept (its theory), state of the art (its history), and advantages and disadvantages (its utility).

CONCEPT

First of all, what is generalization in programming? Consider a one time program (written for only one run) used to update a field in every record of a file. Suppose now that one month after this program had been written and run it was found that another field needed to be updated. The original program is modified to handle the second one-time request. After 6 months the programmer finds that his original one-time update program has been used 6 times each time to update a different field. In order to save himself time in the future if nothing else he would modify his program to accept a control card. On this card would be the size and location of the field to be updated. Then every time his program is run he includes this parameter card in front of the updating data. No more modifications are required to the program. An this is then the first step in generalization.

The generalized approach in programming reduces the number of programs required at an installation. Similar repetitive programming requests are handled with one generalized program and several parameter

cards for each separate run.

Generalization in programming can be applied to a certain class of problems. In the field of file processing, common to all data processing installations there are similar procedures. They have been classified as File Management Procedures. Generalization is readily applicable to this common data processing job.

> File Management includes the functions of file creation, file maintenance, data retrieval, sorting and report preparation. File creation covers setting up a new file and/or modifying the structure of an existing file. It also includes the handling of input transaction messages to create a transaction file. File maintenance and updating includes inserting, deleting, and changing either records or fields within a file. Data Retrieval involves the selection of records (or desired portions of records) from a file depending upon a set of selection criteria. The sorting function includes sorting transactions into file sequence as well as sorting report items into report sequence. Finally, the report preparation function includes editing the report items into the prescribed format, providing column and report headings, counting records, performing arithmetic on records, and perhaps even more complex operations on the records (2).

These procedures are common to the whole computing industry and they are very time consuming.

> A study of what kind of effort was being expended on an electronic data processor while performing data processing showed that less than 20% of running time was spent in performing arithmetic operations. The rest of the time (over 80%) was spent in manipulating data to maintain files, sort records and print reports (16).

File Management is a natural for generalization. The need is there. Is it possible to develop a generalized system to meet this need? In such a system a few generalized programs would be all that is required for the whole job of file management. Every application could be solved by adding parameter cards to make specific programs for each job. Attempts are being made to generalize the problem of file manage-

ment as will be seen under the heading "State of the Art".

However, this thesis is concerned not with the broad aspect of File Management but with one phase of that concept: generalized data retrieval and report preparation. It is concerned with the problem of developing a single program which will answer all requests for data from files and prepare different report formats as required. The following discussion, therefore, concerns the concept of generalization as it relates specifically to data retrieval.

A generalized program can be compared to a compiler (16). In both cases input to the program is parameter cards in a pseudo-language related closely to a broad classification of problems. The compiler and generalized program both produce as output a specific program in the language of the particular machine. However, the structure of the machine program is different in each case. The compiler program can produce a completely variable structure. For example, on Fortran program differs completely from another as far as logic flow is concerned. The generalized program on the other hand produces machine programs that are very similar in structure. Their variability is acquired by varying parameters which are supplied to the generalized routine. For example; a generalized report generator produces machine programs that do the following in every program.

(1) Read in a data record.

(2) Do some calculation and logic on it.

(3) Print fields from it into a report.

(4) Return to #1.

Variability between specific applications is achieved by varying fields required from the records, and the format of the report. In every case the machine program has the same logic flow.

What exactly does the pseudo-language of the generalized program look like and what happens during the running of one of these programs?

Firstly, the language is closely related to the problem to be solved. In the case of the data retrieval program, the language is set up conveniently to explain the format of the input files, the fields required, and the format of the resulting report. Preprinted sheets are used for describing:

(1) The input file organization.

(2) The fields required to be retrieved from the records in the file.

(3) The conditions that have to be met in order that a field be printed out.

(4) The exact format of the resulting report including headings, spacing and summaries.

This is a rather significant deviation from the classical approach to programming. In the classical approach, it is customary to describe a process which will lead to a certain end result. The approach used in the generalized routine is to describe only the end result. The process to be used in achieving the end result is determined by the generalized routines (16).

Secondly, with reference to figure I, the logic flow for using the generalized routine is as follows:

(1) The programmer prepares the parameters for his retrieval problem on reprinted sheets.

(2) These are keypunched and run through the G.D.R. program which produces an object deck (or stores the program on auxiliary storage until it is called for).

(3) The object deck is combined with the input files, data is retrieved, and a report is produced.

This defines the concept; the following section will describe how well this concept is known.

STATE OF THE ART

Generalized data retrieval programs have had quite a history as far as the computer age is concerned. The first published paper on such programs written by any user was in mid 1957 (16). This was just 6 years after commercial data processing came on the market. They were developed because the need was there. At that early date the savings in time and cost achieved by generalization were recognized by industry. As knowledge of the concept widened several industries combined their efforts to produce generalized programs of broader scope. One of the better known systems was 9 PAC developed by the combined effort of 12 major computer users and maintained by I.B.M. applied programming group.

The second generation computers provided a more powerful language capability than those on which generalization was first attempted. This resulted in a few more widely available systems but mainly in individual user systems tailored for their particular needs. These companies made available their systems for sale to other companies whose computer was compatable. The generalized systems involved a

great deal of programming effort to develop and obviously the corporations that produced them were reluctant to offer them freely to the computer industry as a whole. This phenomena has hindered industry-wide acceptance of generalized programming somewhat.

To illustrate the impact of the generalized file maintenance and report preparation approach to programming, here is a list of some corporations who have devloped their own systems.

(1) The Naval Ordnance Laboratory, Corona, California. They have devloped a set of 20 generalized programs for handling files of information which are sorted on magnetic tape. With these generalized programs, they have been able to handle just about every data processing request with efficiency and dispatch. When they first set up this generalized system, they were maintaining three data files on magnetic tape. Today they are maintaining some 30 active files. But in the face of this ten fold growth in workload, their programming staff today is smaller than it was four years ago. When they are requested to set up a new data file the systems and programming time required for programming and debugging all of the runs (input runs, sorts, file maintenance, report preparation) usually amounts to something like two man weeks. When they get a request for a special report to be developed from one of the files (and they get a number of such requests every day), the program for the general report generally is developed in from one

to two hours. Their special reports can involve quite complex selection criteria, for the selection of information that will appear in the reports.

(2) Advanced Information Systems Department of Informatics Inc., Los Angeles, California.

(3) Systems Development Corporation, Santa Monica, California.

(4) Smith Kline and French Laboratories, Philadelphia, Pennsylvania.

(5) Imperial Oil Limited, Calgary, Alberta.

(6) General Precision Corporation.

(7) Standard Oil of San Francisco.

(8) National Cash Register Corporation.

(9) General Electric Corporation.

(10) Humble Oil and Refining Company, Baton Rouge, Louisiana.

(11) XEROX Corporation, Rochester, N.Y.

(12) Westinghouse Electric Corporation, Pittsburg, Pennsylvania.

(13) Dupont, Wilmington, Delaware.

(14) Consolidated Edison, N.Y.

(15) John Hopkins University.

Finally, to bring the discussion up to the present: the computer manufacturers are offering generalized report preparation software for their third generation computers:

(1) International Business Machines for their system /360.

(2) Honeywell for their Series 200.

(3) General Electric for their Compatables /400.

(4) Burroughs for their B200/300 series.

(5) Univac for their 1004, 1005, 1050.

(6) Control Data Canada for their 3000 series.

(7) National Cash for their NCR 315.

(8) R.C.A. for their Spectra 70.

Indeed this concept must be significant when I.B.M. offers a generalized programming system that will be used exclusively on their system /360 model 20.

ADVANTAGES AND DISADVANTAGES

Here is what the U.S. Naval Ordnance Laboratory, Corona, California says about their generalized programming system mentioned above.

> Although there are disadvantages, and some serious ones, in utilizing a general approach to information processing, we are convinced that it has been the right direction for us and we recommend it as the right approach specially for any information processing group which has a limited programming effort staff and many customers to satisfy. It has permitted us to devote our programming effort to the continued expansion of our capabilities to the point where our current complex of processor programs provides a really extensive, flexible, and powerful system available for the use of all our files. This would never have been possible had we taken a tailored approach to systems design (14).

Some specific advantages are:

(1) The generalized report generators are most suitable for one time special reports where fast programming is required. The one time special reports that management often requests from the data processing department have been ill received by programmers and management because of the long time they take to produce. So management has kept away from them.

However, with the advent of popular generalized routines one time reports can be easily handled. In fact the generalized report program may offer a breakthrough in management reporting requirements. If he is able to receive his one time special studies quickly he doesn't need the repetitive reports. His decisions can be made more confidently from small one time predictive reports than they can from a mass of repetitive descriptive reports. Management recognizes this point. Until recently they haven't been able to realize it in practice. Generalized Data Retrieval, it is hoped, will be the key to better management reporting.

(2) Novice programmers can easily use the generalized program since its input parameters are problem oriented. Programming effort is costly in time and money and programmers require a lot of training. This is the classical attitude. It can be changed if generalized programming really comes into its own. With the report generators provided by the manufacturers in their third generation of computers the thin edge of the wedge has been struck. Their acceptance is assured and their devlopment inevitable.

Finally here are some of the disadvantages encountered in using the generalized approach.

(1) If has an inflexible information organization (i.e. file organization). Because they are adapted to common data

processing problems the generalized routines do not provide for unusual file organizations. The second part of this paper will discuss such a problem and its solution.

(2) The pseudo-language of the system needs to be improved to facilitate programming more complex File Management problems. This is especially true as real time computing becomes more popular.

(3) It is not suitable for special cases in real time computing where remote terminals will supply both requests for data and raw data to be stored in the computer.

(4) There is probably a sacrifice in efficiency over the tailor-made program. In some cases the running time of a tailor-made program will be quicker than the same problem programmed for the generalized routine. This is offset by the ease of programming for the generalized routine.

(5) Any general easy-to-apply information processing technique is likely to encourage the proliferation of inadequately planned, ill conceived information projects. This is to be guarded against (14).

What has been described so far is a conceptual look at generalized programming, in particular, generalized data retrieval. What is to follow is a detailed look at an example of the generalized data retrieval approach, the I.B.M. Report Program Generator for the System /360.

CHAPTER III

A PARTICULAR EXAMPLE -

SYSTEM /360 REPORT PROGRAM GENERATOR (RPG)

DEVELOPMENT AND APPLICATION

The Report Program Generator is a computer language devloped by the IBM Corporation for their /360 computer. It isn't a unique language (all of IBM competitors have devloped similar languages) but it ranks high in comparison to others (as will be seen later in this chapter). Its scope isn't broad but for certain applications it is very powerful. So far it hasn't been widely accepted but will become more useful in the management information system environment.

RPG is a generalized data retrieval and report generator program. It is an example of the concept discussed in the previous chapter. It is generalized because it has the ability to retrieve data and print reports of all sorts depending simply on the alteration of control cards. The first part of this chapter will discuss these control cards; the way in which RPG converts them into a program, and the range of possibilities with these control cards.

First of all, RPG is a language translator. It uses control cards as input to generate a complete object program separate from itself. This object program can be run immediately or it can be put on auxiliary storage for later use. It doesn't use control cards to alter itself and become a specific program as some generalized routines do. It is a true language translator. Input to it is a problem oriented language,

output is a specific object program.

The control cards are of four main types:

(1) File description: describe the file organization and storage location of the input file.

(2) Record description: describe the location and type of fields on each input record.

(3) Calculation description: describe any calculations required with the fields.

(4) Report description: describe the format of the printed report.

There are actually six card types RPG will handle. Their format is fixed (see figures I, II, III) and predetermined. They are prepared on preprinted 80 column specification sheets, keypunched in the proper sequence and put into the computer. RPG translates these cards into an object program which retrieves the data desired and prints the required report. The various types of cards are:

(1) File description; used to describe the organization and location of the file to be searched.

(2) File extension; used to describe any tables that might be used to assist in accessing the records of the file.

(3) Line counter: used to arrange temporary storage of the report for later printing.

(4) Input: used to describe the location and type of fields by name on each record of the file.

(5) Calculation: used to describe any logic and calculation

desired on the input record fields.

(6) Output: used to describe the format of the resulting report: heading, location of fields from the input file and totals.

A complete description of the specification sheets is available in the RPG manual. Included next are some of the outstanding features of RPG.

(1) It will retrieve data from fixed format records efficiently. Each field in the record must be defined as to its relative position and length, no variability is allowed here.

(2) The input records may come from any data file on cards, tape or disk. On disk the file may be organized in sequential, index sequential or direct.

(3) All or just a few records may be retrieved using RPG. The latter is accomplished by a table look-up of the KEYS of the record to be retrieved.

(4) Arithmetic and logical comparisons are possible on an easy to program level. The source statements are simple enough to be handled by novice programmer although they do not compile into very efficient machine code.

(5) Printed reports are produced by the simple programming of headings, input records and totals.

RPG has been designed as a simple to program language which will handle normal reporting requests from straight-forward simply organized data files. It is not oriented to complicated data processing at all.

COMPARISON TO OTHER REPORT GENERATORS

How then does this generalized retrieval program stock up against those written by other people? Very well. It is the best of all similar systems studied in its category (i.e. manufacturer written report generators). It isn't as powerful as user written report generators. However, the latter are written for specific applications and are therefore more powerful in some concepts which are important to their company. RPG it must be remembered is an industry wide generalized program. It must handle the largest number of report requests. These are relatively simple requests and so RPG has been written to handle these quite well. It doesn't handle any complex files as these are peculiar to only a few organizations. It must sacrifice some versatility for efficiency to keep its compiling time minimum.

In comparison to other manufacturer written report generators, RPG comes out on top. Firstly, it appears to be the simplest to understand for novice programmers. Both manuals put out by Honeywell (for their Series 200) and General Electric (for their compatibles /400) are more difficult to read and to write programs from. Secondly, RPG compiles in one pass, while the Honeywell Report Generator requires two passes through the computer to produce an object program. Thirdly, RPG offers a wider and more refined set of extra features in its language than the other two mentioned. On the whole RPG seems to be a more advanced programming system than similar systems studied from other manufacturers.

As mentioned before RPG doesn't compare too favourably with user

written systems. An the reason: RPG is industry-wide while the user programs are specifically applied to their own applications.

Some comments received by corporations who have developed their own systems show some of the short-comings of RPG. The Humble Oil and Refining Company of Baton Rouge, Louisiana had this to say about their own retrieval and reporting system: "During the run, data from the records are picked up and processed in accordance with prewritten specifications, control fields, summarization, output control stage generation, etc. All of the collections and resultant output records needed to produce all of the desired reports are handled on one pass of the input file. Then the sorted collection output records are fed to the report phase of the system, where prewritten report specifications control the generation of the desired reports. In our Sales Project the monthly collection and reporting runs produce 25 to 30 different reports, all in one pass. This differs from report generating systems such as IBM's Report Program Generator, where the collection and reporting specifications for each report are compiled into a run that produces only that report, not multiple ones."

Here they were interested in developing a generalized data retrieval and reporting program which could be used to produce reports from their daily operation. RPG is more oriented to "one time" report requirements.

XEROX corporation have developed a similar system where the retrieval phase is kept separate from the report phase. Imperial Oil Calgary has developed an advanced retrieval system but it is oriented

specifically to their own files (of oil well information). Their own people in Toronto are unable to use this system because their retrieval problems involve accounting and not well data information.

RPG compares favourably to all the user systems studied. It is of course a different generalized program, developed to do a different job than most user systems. It serves the industries just starting in Data Processing who as yet have uncomplicated storage and retrieval problems. RPG isn't highly refined nor really industry encompassing. It has real application at its own level. The following section will illustrate what this level is and how well RPG meets it.

USER'S EXPERIENCE

Their overall impressions are well summarized by Mr. B. N. Munro, of Alberta Government Telephones who, in correspondence with the author, said: "We will continue to regard RPG as a valuable tool for the rapid mechanization of relatively uncomplicated low-volume applications." It is at this level that RPG is most useful. The users who were questioned all mentioned the following facts about RPG.

(1) Easy to learn.

(2) Easy to write.

(3) Simple to debug.

(4) Rapid compilation.

(5) Little documentation required.

These comments show RPG to be ideal for novice programmers who are given simple programs to get experience on.

The following comments suggest its limitations:

(1) "Our programmers invariably express a preference for assembly language programming because there is no doubt that a more direct machine language approach produces the highest degree of efficiency and utilization of the full potential of the equipment." Maurice Head, Canadian Wheat Board, Winnipeg.

(2) "An easy RPG program is easy in any language. A difficult RPG program is average in Basic Assembler. A difficult basic assembler program is probably impossible in RPG," L. F. Gordon, Ohio State Life Assurance Co.

(3) "As the complexity of the program increases the running time, core usage and difficulty of de-bugging increase to the point where using RPG is no longer economically feasible," A. T. Hirsch, J. L. Hudson Company, Detroit.

(4) "RPG does not provide for modularity (breaking a program into sub programs) and it is difficult to patch object programs to any significant extent," B. N. Munro, Alberta Government Telephones.

Again the comments show RPG to be suitable for novice programmers and simple jobs.

It is also interesting to note that several companies who have installed /360's have as yet not used RPG at all. The Prudential Insurance Company of Americal has installed "several" /360's but have had "no significant experience with RPG". This indicates that the RPG pro-

gramming system isn't very powerful on large scale operations. It is more suited for simple jobs.

All the above comments suggest that RPG isn't the answer to data retrieval and reporting in data processing today. In fact RPG seems to be just another fancy language with little application. Is this really true? No. As yet users experience is limited. The potential of the RPG type program isn't recognized. Industry isn't ready to progress from repetitive reporting to one time decision making reports. RPG is really a stepping stone in the direction of a management predictive reporting language. This also is the concept of generalized data retrieval. Its utility isn't for large accounts receivable applications but for simple data retrieval and report preparation programs of a one time nature. It finds its use not where complex logic and arithmetic are required but where simple the status of a record is queried. RPG solves simple requests for data to be presented in an organized fashion, the interpretation of which is done by the person who reads the report. The problem solved in the subsequent part of this thesis uses RPG for the latter type of report requirements.

PART II

AN APPLICATION FOR G.D.R.

CHAPTER I

THE TIME SEQUENCE FILE

REASON FOR THIS FILE

There exists today in EDP a very restrictive phenomena; the limitation of fixed and variable format file organizations. The fixed format of the 80 column card posed a problem in the days of unit record equipment. More recently file designers have encountered the problem of fitting their source information into fixed and variable format tape and disk records. The location of information within these records must be predetermined. This is a normal procedure. The program must know where to find a field in a record and it must know how large it is. But their conditions are restricting. In many instances certain fields may not exist for every record or may be of variable length when they do occur. The file designer usually allots space in every record for every possible field and this is a waste. Or he uses control fields to specify the existence and length of these fields. Again there is a waste of space.

The limitation on the type of file that the designer has to work with becomes more significant as the real time computing develops. Today the computing industry with the aid of third generation computers is attempting to reorganize their data processing. They would like the source information to be read directly into the computer and stored as it comes from the envirorment. Fixed and variable length formats do not lend themselves to this type of computing. They demand that similar

information entering the computer at different times be classified separately but remain exactly the same length and format. Today computing problems require more freedom. The time sequence file is an attempt at providing this freedom.

There is a trend today towards generalization in order to make life easier for the programmer. A rule of thumb says that he is equally as expensive as the initial layout for the computer. So it is essential that his time is well spent. He has been assisted by compiler programs like Cobol but he still struggles with the problem of file organization that must be continually revised and updated. Why shouldn't a generalized approach be taken to this latter problem too? The time sequence file is an attempt to generalize.

A time sequence file or unformatted file is one in which data can be added quite conveniently. It does not require a predetermined location for the information. Several similar fields entering the computer at different times are simply packed onto the end of the record in order of their appearance. The first information appears first, the next in time comes next and so on until the record is full.

But enough for general introduction, now what does the time sequence file actually look like?

HOW IS IT ORGANIZED

The details of the time sequence file have been studied by Tiwari in his paper on file organization (21). However, for the purpose of this discussion a resume is provided. The application of this file to

EDP will be explained next, followed at the end of this paper by several examples of its use.

The following explanation will be more easily understood with the aid of figure IV. The time sequence file as its name implies is organized on the basis of time. The records that make up the file are created as information comes from the environment. They are deleted when they are no longer active in the environment. For example: when a student enters the university for the first time he is given a number and a record is created in the student file on disk. This record follows the one for the student who was one ahead of him in the registration queue. When the student leaves the university permanently his record is removed from the file. It is no longer active. It is stored on tape and will probably be referred to only a few times in the rest of the student's life time.

Each record when it is created has a fixed length and every record will be the same. This length is optimum (it depends upon the application). It is decided upon by a statistical study of the amount of information that will be in the record over its active life time in the file. It should be long enough to accept data from 95% of all cases (students). This means that 95% of the records will not be filled or just filled with information and only 5% of the records will be overflowed. The overflow principle will be explained later.

The record is made up of two portions:

(1) One main logical record (MLR).

(2) Many secondary logical records (SLR).

The MLR has the following characteristics:

(1) The MLR is established when the record is created (when the student registers for his first year).

(2) It occurs only once, at the beginning of the whole record.

(3) It contains a key field (student registration number) by which the record is retrieved and other information that is determined from the environment at this time (student name, address, date of birth).

(4) The format of the MLR is fixed. Each record in the file has an identical MLR format.

(5) When the record is created a length of disc storage is reserved and in the first portion of this length the MLR is stored.

Now, during the life of the record (time student is in university) information about the student enters the computer and must be stored on the record. Each separate message (student course numbers, student exam mark, professor comments) from the environment is stored in the record as a separate sub-record called an SLR. These sub-records form the most important part of the record. They contain all the accumulated information about the record. These SLR's have the following characteristics:

(1) They contain one or several fields related to the message from the environment.

(2) They are formatted identically for each occurence of a similar message.

(3) Each SLR has an identification code which classifies its format and information content.

(4) Similar SLR's (same identification code) are chained together within the record for ease of retrieval.

(5) The first SLR of a certain identification code that occurs in the record is chained to a table in the MLR which also assists in retrieval. The actual mechanism of chaining and retrieval will be better explained in the examples.

To return to the problem of overflow where 5% of the records are not long enough to contain information about the student. This condition may occur when a student has an unusually long stay in the university. An overflow occurs when information coming into a record from the environment finds that the record is completely full. This information then goes to an overflow area, which is a separate area of disk storage from the main time sequence file. The main record and overflow are chained together for retrieval purposes. The two are treated as one record.

To summarize on the time sequence file organization:

(1) A record is created and initial information enters it in the MLR portion.

(2) The SLR's fill up the rest of the record as messages which create them enter the computer.

(3) There is no fixed location within the record for these SLR's. They may appear anywhere or not at all.

(4) They are easily entered in the record. The next message is

butted up against its predecessor in time sequence.

(5) If too many messages occur the excess is put in an overflow area, and the main record is chained to this overflow area.

WHAT IS ITS APPLICATION

The time sequence file is a type of file organization which will have wider application as real time computing develops. The information from a real time environment is first edited then stored on the file. Storage of this information occurs in the first free space within the record without regard for predetermined fixed locations. The storage of all messages is identical. This means that programs for putting data on the file are identical. And they can be quite short. In addition, information that is discovered after the file is set up and programs written is added with very little additional programming. Modification of existing programs is eliminated. Thus the technique lends itself to the real time problem of changes that must be effected immediately.

The advantage of this file over the fixed predetermined format is therefore obvious. A new SLR may be created and entered into records of the file without disturbing the complexity of programs that process the file for other applications. The third generation of computers has made available a new base for computing (large random access storage). It is up to the users to utilize this base to its fullest by developing new concepts in software, (generalized programming). The time sequence file is an attempt to make the best use of the computing base and ad-

vances in software available.

It has been mentioned that information can be put on the time sequence file very simply. This is accomplished by a generalized file maintenance program. However, retrieval from this file is another problem altogether. The remainder of this thesis is devoted to a study of data retrieval from the time sequence file.

CHAPTER II

RPG AND THE TIME SEQUENCE FILE

To review, the discussion thus far may be outlined as follows:

Generalized data retrieval as a programming base has been explained on the conceptual level. RPG as working example of generalized data retrieval has been explained as it applied to fixed format files. Finally a new idea in file design has been presented. It now remains to attempt to modify the generalized data retrieval program (RPG for /360) to handle this unique file organization. Two separate attempts were made at this modification.

(1) Internal - change some coding of the RPG language translator program.

(2) External - add an assembler S/R to the RPG object deck.

MODIFICATION REQUIRED TO RPG

The RPG program (language translator) as has been explained earlier is able to retrieve data from disk files according to the three main organizations (sequential, index sequential or direct) but only from fixed format records. The location of fields within the record must be specified on the input specifications. The RPG program generates object code which takes each field from the input record and stores it in a SLOT. One slot is generated for each field appearing in the input specifications. Subsequently the program handles any request for a field by referring to the necessary slot. For example when a field

is named on the output specifications for printing the RPG program generates object code that moves the field from its slot to the print area. So the object code generated is specific. It knows where the field on the input records is located and the address of the slot.

In the time sequence file the location of the fields within the record cannot be predefined on input specifications. SLR data fields may occur anywhere in the whole record. They are not in fixed order. It appeared therefore that a modification to RPG would be required which would allow the fields in the input record to be located, and put in the required slot when the program is run. Two separate techniques were studied for locating the fields within the record:

(1) Search the whole record for the desired field (scan the record SLR by SLR) on an identification code match.

(2) Use a table look-up available in each record (MLR portion) that located the position of the desired SLR within the records.

These two were studied by Tiwari. (21) His investigation found the table look-up approach more efficient. More will be explained in the examples about the table look-up idea.

A second modification was required because of the repetitive nature of the information in the record. Information in this time sequence file is not updated, it is accumulated for the whole life of the record. For example repeated attempts at an examination are always recorded, not just the final pass mark. Thus several SLR's of the same type may be present in the record and retrieval would require accessi-

bility to them all. That is, it may be necessary to prepare a report of all the marks a student obtained in his attempts to pass a certain course. This implies that looping capability is required in the retrieval of information from the time sequence file. The start of a loop is required at the point where the retrieval of the first time sequence SLR of a certain type occurs. When the SLR is located the program puts its fields into slots. Then this information is printed. Finally a branch must occur back to the point in the program where the first information was retrieved. This allows the same coding to look for the second, and subsequent SLR's of the same type. The procedure continues until all SLR's that satisfied the programmed requirements are removed from the record. Then the next record in the file is retrieved and the procedure repeated. The RPG program does not provide the facility for looping back from output specification to input specifications on the same record. Every time the program returns to input specifications it is to retrieve another record. Thus it was concluded that a second modification was required in the RPG compiler program. It would allow for generation of code for branching from printing back to searching the input record for more information.

Two modifications only were required to handle time sequence files with RPG. Conceptually they were defined. But practically it still remained a problem to locate the correct spot for their inclusion. If they were to be internal modifications where should the compiler coding be altered and exactly how should it be altered? If they were to be external modifications where should the assembler subroutine be

added to the RPG object code, how should the two programs be linked together and exactly what routine should be written? The answers to these questions are given next.

INTERNAL ATTEMPT AT SOLUTION

The first attempt at solution was to alter the RPG language translator program. If this could be accomplished then the programmer working with time sequence files would have a relatively easy time. He would simply specify the SLR identification code and fields he wanted retrieved and RPG would take care of generating the object code required. As a generalized data retrieval solution the above procedure was very desirable; but was it feasible?

A listing of the RPG (Tape Operating System) program was obtained and later the RPG logic manual. They had the following specifications:

RPG (TOS) source listing

(1) 690 pages

(2) 41000 lines of programming

(3) 25 programs

RPG (BOS) logic manual

(1) 350 pages of flow diagrams

(2) 408 pages altogether

It took 160 man hours of study to narrow down the location of the first modification to one program. A spot for the second modification, the branching routine, could not be determined. The language translator program was simply too complex to break down within the alloted

time. Even with the additional help of the logic manual the exact spots could not be located.

The concept in software with the /360 is to write programs in sections then link them together for execution as one program. This was the technique used in developing RPG. Various sections were written separately then linked together. The sections themselves were complex but logical linking was simply too difficult to follow without assistance from the original program designers. This latter alternative was rejected as uneconomical in time and money.

Having reached a dead end on the approach of compiler program modification the second alternative was chosen. It fitted more easily into the existing framework of RPG and proved successful as will be seen from the examples.

EXTERNAL ATTEMPT AT SOLUTION

The problem again, was to modify the RPG program to handle the time sequence file organization. The attempt at solution involved adding an assembler language subroutine to the object program generated by RPG and running them as one program. It was successful.

RPG has included in its logic the capability of linking to an assembler subroutine and re-entering following executing of the subroutine. (Refer to the RPG manual under calculation specifications.). The linkage program step may be included after the input record is retrieved but before any output printing occurs (on the calculation specification sheet). It is used normally during "calculations" to

take care of a special type of calculation (e.g. square root) that is not available in the RPG calculation macros. The programmer specifies the EXIT word, and the variable names that he generates in the assembly routine and wants upon returning to the RPG program. When a retrieval job is to be run the CPU in executing the RPG object programs does the following:

(1) Starts by reading an input record

(2) Branches to the subroutine.

(3) Executes the assembler written instructions, which perform some logic and arithmetic on the RPG input record fields.

(4) Branches back to the RPG object program with the resulting fields available for printing of the report.

(5) Prints the report for that input record.

(6) Goes back to #1.

In #4 fields may be generated by the assembly routine and these can be taken back to the RPG program if required.

The solution to retrieval from the time sequence file using the EXIT facility of RPG was as follows:

(1) The time sequence record was defined on the file description specifications and to a limited extent on the input specifications. The RPG compiler generated code that read a whole record into core from the disk. The whole record (containing yet undefined SLR's) was broken up into convenient length slots. These slots were in contiguous storage so they could be treated as one continuous record as it appeared in the

disk.

(2) The EXIT statement was used to send the CPU to a subroutine written in assembly language.

(3) The assembly subroutine searched the whole record for the required retrieval fields and these were then sent back to the RPG program (the table look-up searching routine will be explained in the examples).

(4) The assembler routine returned to the RPG program at a location following the EXIT statement.

(5) The RPG program took care of printing the retrieved fields.

(6) The RPG program then branched back to the beginning where a new record was read in from the disk and the procedure from #2, explained above, repeated.

This was the solution attempted and completed. Immediately several conclusions may be drawn from the above.

(1) A Generalized Data Retrieval program is already available to /360 users to retrieve the whole time sequence file from disk and print a report from it.

(2) The fields to be printed from the time sequence record are specified and retrieved by a specific assembler routine. The input file and resulting report formats are described as parameters in specification sheets for RPG. But the fields that are to be retrieved by the program are described in program instructions in the assembler subroutine. They are not presented to the subroutine as parameters.

One assembler routine and one RPG program must be written for each retrieval application. The retrieval technique will be identical in each case. For the same file the changes from program to program will be:

(i) RPG output specifications

(ii) Assembler subroutine addressing of desired fields.

(3) The two modifications explained at the beginning of this chapter were reduced to one assembler routine. All SLR's of a similar type were retrieved in this one subroutine. This will become more obvious in the examples.

The assembly subroutine addition to RPG proved successful in retrieving data from the time sequence file. The following examples will illustrate this statement.

CHAPTER III

EXAMPLES

The following examples are intended to serve as illustration for the principle just explained. They illustrate the approach to applying the time sequence file to RPG. There are four examples. Each one answers a request for a different type of report. Although individually they increase in complexity, collectively they demonstrate one technique. This common approach is as follows: (see figure V).

(1) The file (the time sequence file) is described on file description specifications as a fixed format disk input file. RPG takes care of all the IOCS required to access this file.

(2) On input specs the MLR fields of the records are described.

(3) On calculation specs an EXIT is used to branch to a user subroutine.

(4) In the subroutine the fields required to be retrieved are searched for using a table look-up and chaining technique.

(5) A branch occurs back to the RPG program.

(6) The fields retrieved in the assembly subroutine are printed on a report described on output specifications.

(7) The RPG program returns to the time sequence file and retrieves the next record in succession.

This is the overall technique. Now to explain the detailed retrieval approach in the user subroutine (assembler routine) (see

figure IV).

(1) Certain control fields are available within each record of the file which assists in the searching.

(i) Each SLR has an identification code and each repetition of the SLR has the same identification code.

(ii) Each SLR has the record KEY code for validity checking during retrieval.

(iii) A table is set up in the MLR portion when the record is created containing x and y entries. The x entries compare all SLR identification codes which may occur in the record. The y entry is a relative address of the first occurance of that SLR (defined by the corresponding identification code) within the record. This address is relative to position one at the beginning of the record.

(iv) Each SLR has in addition to its identification code a relative address which indicates the location within the record of the next similar SLR.

(v) Each record is defined according to its KEY which appears at the beginning of the MLR portion of the record.

(2) Backing up for a moment, using the RPG program the records are retrieved by KEY.

(3) Once the record is in the memory a branch occurs to the assembler subroutine.

(4) The assembler subroutine is written as a specific program. It does the following:

(i) Locates the relative address in the table corresponding to the identification code of the SLR to be retrieved.

(ii) Uses this relative address and the record's absolute address to locate the SLR.

(iii) Puts the fields to be retrieved from the SLR into slots.

(iv) Looks at the relative address (chaining address) within the SLR.

(v) If this address is not zero the next SLR address is formed (adding the record's absolute address and the next SLR relative address). The next SLR fields are put into slots and the program returns to iv.

(vi) If the address is zero no more SLR's of the same type occur within the record and a branch occurs back to the RPG program for printing the slots for this record.

(5) The printing is done, another record is retrieved and the searching begins again.

To review for a moment, why do repetitive SLR's of the same type occur, and require connection? An illustration will answer this question. Successive attempts by a student to pass a certain course are each recorded in the student's time sequence file. Each attempt is recorded in sub-records of the identical format (SLR with the same identification code). However, they do not appear in consecutive order within the record. Between each attempt at this course there may be records of other exams passed or failed, of fees paid or of student activities. In order that the results of all attempts at this exam be

retrieved quickly these SLR's are chained together. Thus repetitive entries of the same SLR format exist and must be logically connected for retrieval. This is the basic concept of the time sequence file organization. The retrieval of these SLR's is accomplished by using RPG and an assembler subroutine.

The above has been an outline of the detailed approach used to combine the time sequence file and RPG. The following examples will use this approach.

EXAMPLE I

Each example is made from four listings, L1, L2, L3 and L4 (see appendixes I, II, III and IV). The first listings is that of a time sequence file on disk. It is listed in 80 character lines. The second listing is the RPG source program, and the third is the assembler subroutine. The final listing is the report produced from the time sequence file by the combination of RPG and assembler program.

Example I illustrates very simple report requirements. Here is a statement of the problem: Using the disk file MEDRE, search every record for field A in SLR 02. Print out the first one in every record that is less than 11654.23. Accumulate all these and print out the sum at the end of the report.

The file MEDRE is located on disk. It is a fixed format, sequential file of 400 character records. Each record is organized in time sequence order. It appears in L1 appendix I in 80 position groups. Its five records have the format of figure VII and the required SLR (identification code 02) has the format of figure VIII. Looking at the first record (that of J. B. MITCHEL) the table indicates that SLR's 01, 03, and 04 do not appear but that of SLR 02 occurs in position 80. This is the first occurance of SLR 02. The SLR key and identification code match to the MLR key and table code (02) respectively. However field A is greater than 11654.23, so it isn't printed. The next SLR to be tested appears in position 160 and the next in 240. Finally in position 320 of the record is an SLR that meets the comparison requirements

(347.65 is less than 11654.23). This field is therefore printed and the program goes on to the next record. This SLR is also the last in the record (chain field is zero) therefore the program has to pass onto the next record. For convenience in this example each record has similar SLR's. Looking now at L4 of appendix I the required report appears. Each record in the file is presented by name and address. The field A meeting the comparison requirements is printed and summarized at the bottom of the report. Only the A fields meeting the requirements are listed although all A's are located in the record.

Listings L2 and L3 represent the programming required to obtain the report in L4 from the disk file listed in L1. L2 is the RPG program. One input file (the time sequence disk file MEDRE) and one output file (the report) are defined. On input specs the MLR fields are defined for the input file (i.e. KEY, X Y TAB (x y table), NAME, ADDR (address). The rest of the record must be defined in sections not greater than 256 characters. For simplicity the whole record is defined again in sections of 200 characters. This provides slots for the whole record in working storage. The slots are consecutive so that the record appears in contiguous order in working storage.

On the calculation specifications a small logic program is developed (see L2 appendix I). A branch occurs to the subroutine (EXIT GTSLR1) which locates and puts into a slot the first A of a record. A branch occurs back to the RPG program. In the RPG program the field A is compared to 11654.23. If it doesn't meet the programmed requirements a branch occurs back to the assembler subroutine (GOTO LOOP and the

(EXIT GTSLR1) where another field A is searched for in the record. Finally a field A meets the RPG comparison requirements and it is summarized in field SUM. Printing then is done of the KEY, NAME, ADDR, and field A. The RPG program branches back to retrieve another record from the disk and the procedure repeats itself. When all records are retrieved the field SUM is printed (on last record).

Also on the calculation specifications each variable required in the assembler routine appears under RLABL. REC field must appear in order that the address of the whole record is sent to the assembler subroutine. The field retrieved from the assembler searching routine is A (ULABL A). Indicator 02 on indicates a field A has been located in the assembler routine, and 06 on indicates that it meets the comparison requirements, so it can be added into SUM. If indicator 07 is turned on then the requirements are not met and another search must be made in the assembler routine.

On the output specifications headings are printed first. When a record is retrieved from the file indicator 01 is turned on and the MLR information is printed. If indicator 02 is on during the printing cycle a field A has been located that meets comparison requirements and can be printed. Finally after the Last Record has been read the SUM field is printed. Note that indicator 03 is an error indicator. It is turned on in the assembler subroutine if a validity check occurs (an invalid SLR KEY code). The program prints an error message and continues to the next record.

The assembler routine is designed to be simple (see L3 appendix

1). All input data is supplied to it by RPG and all output data is taken from it by RPG for printing. No IOCS is required. This subroutine simply searches for the required SLR and moves its fields into SLOTS for the RPG program. Variables coming from the RPG program must be included in an EXTRN statement (for linking the two programs together). Since the record is stored in working storage in contiguous sections only the first section need appear in the EXTRN statement (i.e. REC in this example). At the beginning of the program the registers are stored since their values must be present when control returns to the RPG program. A specific table entry (x=02) is inquired to see if a chaining address (y) occurs for SLR 02. If the y value is zero no chaining address occurs. The registers are restored to their original value and a branch occurs back to the RPG program. If y isn't zero an SLR of type 02 occurs within the record. An actual memory address is generated for the SLR. It is then validity checked and field A is moved into a slot. Indicator 02 is turned on and a branch occurs back to the RPG program. If this field A doesn't meet the comparison requirements the RPG program returns control to the beginning of the assembler subroutine which now generates the address of the next SLR (if one exists) from the chaining address that appeared in the first SLR. Validity checking and moving of A into the slot occurs again. A branch occurs to the RPG program and the process repeats itself. RPG addresses are determined by address constants and the dummy control section. An ENTRY is required for addresses sent to RPG.

When a new record is read from the disk the assembler routine is

reinitialized automatically with the correct control data for searching. Note that indicator 03 is turned on in the case of a validity error and this fact is printed out on the report.

This retrieval logic repeats itself for all the Examples considered with minor variations depending upon the report requirements. However, in one respect Example 1 is a special case. In it the assembler routine may be entered several times for each record retrieved from disk. In the following examples, and as a general rule, the assembler routine is only entered once for each RPG cycle through input, calculation and output. All fields to be retrieved are located and slotted during one EXIT to the subroutine. The subroutine isn't entered again till the next record is to be searched.

The next three examples are of interest not so much for the technique of retrieval as for the type of problem they are solving.

EXAMPLE II

The statement of this reporting requirement is as follows: Using the disk file MEDRE locate and print every occurance of field A in SLR 02 for every record. No more than 12 values will appear in any record. Print the values of A, three to a line. No accumulation is required.

Looking at L1 appendix II it can be seen that 10 SLR's of identification code 02 occur in each record (similar records were used for convenience). They occur in positions 64, 96, 128, 160, 192, 224, 254, 304, 336 and 368. This can be seen from the chaining address. The record format appears in Figure VII and the SLR format (identical to example I) appears in Figure VIII. These SLR's are to be retrieved and field A in each is to be printed. Three A's appear on each line. The resulting report appears in L4 Appendix II.

The programming required here is slightly different from Example 1. <u>All</u> occurances of A are to be retrieved and printed. The RPG program as before must retrieve a record from disk. It must EXIT to the assembler subroutine which now locates and puts into slots all occurances of A. Control is returned to the RPG program which must format these A field's three to a line. The number of occurances of A is variable between records. Therefore a maximum number of slots is allotted and defined in both the subroutine and RPG source programs. The assembler subroutine searches the record for all A fields. It is given control from the RPG program only once for every record read from

disk. When the subroutine is completed all available A fields are slotted and printing can occur. Control returns to the subroutine only when the next record is read in from the disk.

In the assembler routine, the slots are initialized A1 to A12. The SLR's of the record are retrieved as in Example 1. The number retrieved is maintained in register 3. When it reaches multiples of three another indicator is set on. This is done to assist in printing three to a line on the report.

In this example the validity checking routine is sensed on the invalid sub-record. Record 00005 for Richard Stein has been created incorrectly (see L1). The SLR at position 128 (A field is 3333377) has a correct key, however the ID code is invalid (07 not 02). Therefore on the final report a validity error is recognized and printed as seen on L4.

Example 2 illustrates the standard approach - one entrance to the subroutine to locate a variable number of fields. Examples 3 and 4 will present more complex retrieval requirements using this same approach.

EXAMPLE III

It is conceivable that information to be stored in a time sequence file may be purely alphabetic. For example there may be notes a professor has made on interviewing a student for scholarship awards. These notes can be of any length. Another professor interviewing the same student might also want his notes to be saved, also in the same student record. The SLR's that are created to store these notes contain alphabetic description of variable length anywhere from a few words to fifty or more. The problem is, how to retrieve these notes from a record on disk where they are stored.

Firstly, consideration is made as to the width of the resulting printed report. These notes are of variable length but when they are retrieved they cannot be printed more than 132 characters per line (the normal IBM 1442 printer allows 132 print positions). So they should be broken up into fixed length sentences. For this example groups of 64 characters each were chosen (see Figure IX). Therefore, a descriptive paragraph stored in the time sequence record was divided into sentences of 64 characters each (one SLR). When breaking them up in this manner every one is connected to the previous one by a chaining address, so that they can be retrieved together as one paragraph. The first sentence in the paragraph is chained from the xy table (see figure XI).

When a second piece of descriptive information is put into the file at a later date it is broken up as with the first. Each 64 character sentence is chained to the next for retrieval purposes. However,

the first sentence must be chained from the first sentence of the previous paragraph in time sequence. This allows access to be made to all descriptive entries on a certain subject (e.g. scholarship interviews). Only one SLR type is created. Its data field is 64 characters long and contains alphabetic information.

Putting these sentences together for printing occurs by use of the chaining address (B chain) which links all SLR's of a certain identification related to the same paragraph. If other sentences from later paragraphs are to be retrieved they are located by a second chain (A chain) from the first sentence of the first paragraph to the first sentence of the second paragraph. Sentences within the second paragraph are located by chaining as before (B chain).

Consider the file listed on L1 of appendix III. The records in this file have the format of figure VII and the SLR's of interest have the format of Figure IX. There are two chaining addresses and a 64 character length sentence of descriptive material within the SLR. Refer to Figure XI. The x-y table chains to the first SLR of the first paragraph. The A chain in this SLR locates the first SLR of a later time sequence paragraph. And the A chain of the first SLR of the second paragraph locates the first SLR of a third paragraph. Return now to the first SLR of the first paragraph. Within this SLR is a second chaining address, the B chain. Each B chain locates the next SLR within the paragraph that has been broken up into 64 character sentences. The last SLR of the paragraph has zero value B chain. This terminates the retrieval. Only the first SLR of a paragraph has an A

chain with any value other than zero. If only one paragraph of information has been put into the record at the time of retrieval the first SLR A chain will also be zero.

The first record of L1 appendix III will illustrate the above.

(1) Two paragraphs are written in this record:

(i) "If you have built castles in the air your work need not be lost, that is where they should be. Now put the foundations under them."

(ii) "If a man does not keep pace with his companions perhaps it is because he hears a different drummer."

(2) The x-y table indicates that the 03 SLR occurs first at position 80 of the record. The A chain in the first SLR indicates that the next paragraph occurs at position 240. Thus the two paragraphs are chained together.

(3) Within the first paragraph there are two sentences joined by the B chain, indicated by 0160 in the SLR at position 80. Within the second paragraph there are also two sentences jointed by the B chain, indicated by 0320 in the B chain of SLR at position 240.

(4) Each sentence is treated as an identical SLR of identification code 03.

The resulting retrieval report appears in L4 appendix III.

The retrieval technique (see L2 and L3 appendix III) is identical for that of Examples 1 and 2. Indicators 13 and 14 are used by the subroutine to tell the RPG program how many SLR's have been retrieved.

A maximum of six 64 character sentences are allowed by this program. When the B chain is zero the subroutine returns to use the A chain to locate the next paragraph (ENDTXT). Here as in Example 2 a variable number of fields can be printed. There is also included the possibility of a validity error in the SLR.

Example 3 is similar to Example 1 in that it is a special type of retrieval problem. Example 4 is similar to Example 2 because it is a more common type of retrieval problem. The latter retrieval examples will be encountered quite often in data retrieval from the time sequence file.

EXAMPLE IV

Requests for information from the time sequence file will vary widely. One, however, will be more common than all the others. That will be the request to list all information accumulated under a certain SLR identification code. For example, one SLR type may be used to maintain all financial information on the student. In it would be stored the amounts of payments of fees, scholarships, loans, along with category codes and particulars. The complete financial picture on the student could be asked for and obtained by listing the complete history of one SLR in a students' record. This type of request is probably the most common. It is certainly more common than that of Example 1 where a particular field in a SLR had to meet a comparison condition to be printed. Example 4 illustrates the retrieval of a whole SLR history in the record.

With reference to Figure X and L1 of appendix IV, each SLR of identification code 04 has five fields, one of which is alphabetic. In the file there are five SLR's of identification code 04 for Barry Hartry. Each must be retrieved and printed in a report with spacing and decimal points included in the proper place. The final report produced appears on L4 appendix IV. Note that on L1 appendix IV the fifth record Sanne Jensen has six SLR's of type 04 whereas the rest have five. Only five slots were allotted in the program so that an error condition has occurred for Sanne Jensen report listing. The error was programmed as seen from the error note on L4 appendix IV. It will be explained

shortly.

The programming requirements for producing this report are listed on L2 and L3 of appendix IV. It is similar to the previous examples. Up to five SLR's are allowed per record and these are triggered for printing by use of indicators 15, 16, 17, 18 and 19. The setting of these indicators is done by using address constants and register 7.

The fields are located and moved into a MASK around the decimal points. This procedure ensures proper spacing and decimal point alignment. Validity errors are programmer for as before. In addition the possibility of an overflow error is programmed. This error occurs when more SLR's of a certain type have been written into the record than are allowed by the retrieval program. That is (see L1 appendix IV) Sanne Jensen has six SLR's of type 04 but the retrieval programs (L2 and L3) allow only for five. So an overflow condition is printed out in the report (L4 appendix IV). To obtain all possible SLR's the program would have to be modified to allow for more slots. The overflow condition is programmed in the assembler routine under OFERR.

This completes the detailed explanation of the examples. The next chapter will try to point out some advantages and disadvantages of the approach used.

CHAPTER IV

COMMENTS, SUGGESTIONS AND IMPRESSIONS

In Appendix V there is a summary of the programming requirements for implementing the above approach to retrieving data from the time sequence file. This chapter will discuss some of the broader implications of this approach that need be considered when contemplating its utility on a large scale.

COMMENTS

Some additional comments on the time sequence file which haven't been mentioned before are given here.

Generation of the time sequence file is done by programs outside the scope of this thesis. Data from a real time environment is edited by a generalized edit and maintenance program and formatted into the correct record for the time sequence file. It is the duty of this editing program to establish correct formats, identification codes, validity checks and chaining addresses. The design of the exact details of the above is variable and depends upon the application. The broader aspects of the approach to be used has been suggested by Tiwari (21). It is these broader aspects that have been tested in the EXAMPLES of this thesis. The exact form of the file in these EXAMPLES need not be followed nor need the programming be identical. They serve only to demonstrate that the RPG - Assembly sub-routine combination could actually work with the time sequence file. Variations in approach will no

doubt occur as experience with the time sequence file grows. It is hoped that these variations will upgrade the prestige of both RPG and the time sequence file.

In the time sequence file there appears to be a large amount of space consumed by control fields. In a real application involving fifty different types of SLR's an x-y table of considerable length is implied. This is the old problem of the sacrifice of storage for additional retrieval speed. Control fields make processing much more efficient but of course they require space. By using the System /360 disk storage configuration the problem disappears. Using this computer the x-y table can be easily converted to straight binary digits and therefore kept at an economical size. In fact it is preferable to have them in binary since the y values must be used as addresses and therefore must be converted to binary anyway. Also considering again the disk as the auxiliary storage medium the space used by control fields isn't critical unless a very large number of records are in the disk file. This idea and many others are discussed in more detail by Tiwari (21).

The unsuccessful attempt at modifying the RPG language translator deserves comment.

It suggests two things:

(1) Generalized programs are very complex computer systems. They are difficult to write since they must include a very wide range of possible conditions. Advanced programming techniques are essential in developing them. It appears futile to expect novice programmers to attempt any problem

in generalized programming. Any corporation interested in developing a generalized programming system must realize the enormous size of such a project. It involves a great deal of effort by senior programming staff and should not be treated lightly.

(2) The above realization became apparent when the possibility of creating an original generalized data retrieval program was still being studied for this thesis.

In any generalized data retrieval program developed independently most of the capability of RPG would have to be included (file and record description, calculation, and output report formatting). So a program of the magnitude of RPG was in order. It was not feasible to complete such a program in the time allotted.

The most important problem, however, in the development of an original generalized retrieval program is the input/output programming. The system /360 is controlled by a supervisor program which handles all input/output. A parameter fed program as the one suggested above would have to generate machine coding which would link automatically to the IOCS portion of the supervisor system. A complete understanding of the /360 Operating System supervisor and its IOCS phase is required to handle it. Only very experienced /360 programmers could attempt to handle this problem.

SUGGESTIONS

Speaking of the programming involved in this thesis generally, a philosophy should be recognized in order to use the two programming systems efficiently.

The RPG program is being used to establish all the IOCS required while the assembler subroutine performs the searching logic. Each program is efficient in handling its own function. So overlap of functions should be kept to a minimum. That is, the assembler routine should not include any input/output if possible because this is very time consuming in programming. At the same time and for the same reason the RPG program should not attempt to unscramble SLR formats. The most complex functions to be performed are that of input/output so let RPG handle them automatically. The searching logic is simple and concise so let a small assembler routine handle this. Keep the assembler subroutine small since for novice programmers, it is the more difficult to code. Use RPG for calculation (as in EXAMPLE 1) since novice programmers can more easily understand its language. Also work towards identical coding requirements in every assembler routine. Eventually a generalization of this subroutine may be feasible. (See APPENDIX VI).

IMPRESSIONS

In Winnipeg, Professor B. A. Hodson is working on plans for a computer system at the University of Manitoba's 250 bed hospital using a different programming technique. This could be the first operational hospital information system in Canada (20).

Impressions presented here are in view of the fact that the first major application of the time sequence file and generalized programs may be the above hospital system.

First of all the technique suggested in this thesis is practical. Combining RPG (a generalized data retrieval program with a short concise assembler subroutine) solves the retrieval problem of the time sequence file. The basic idea of using a generalized routine to solve all retrieval requests has been adhered to. It appears (as seen in the EXAMPLES) that this approach is feasible on a small to medium scale. On a large scale the approach must be seen in another light. The following points explain the technique's limitations:

(1) For a record of considerable length (one of 2000 characters or greater) the programming becomes bulky. Each 256 character section of the record must be redefined. For a file of records one per track (2000-3000 characters each) this amounts to over 10 definitions in order to establish continuous working storage for the records.

(2) There may be large variability (in a large application) in the number of SLR's of a certain type in each record within the file. For this reason it is difficult to estimate what the maximum number of allocated slots should be when a program is written. Establishing many more slots than will be required to handle most requests, is essential for handling special requests. This wastes storage although it does not significantly increase the complexity of the

programming.

(3) The idea expressed in number (2) is the result of the one time nature of the subroutine. It is entered only once for each record retrieved. Therefore it must search for all fields that meet the logic requirements and slot them. It would be much better if a field could be located, then printed, then another located and printed. However RPG doesn't allow this flexibility. So the problem of establishing the maximum number of slots before the program is run appears every time such a program is written. As a result the program will generally occupy much more storage than will actually be in use. This is a very definite limitation.

(4) In a real time environment the retrieval programs will be prewritten and stored on auxiliary storage until required. If a large number of SLR types exist, many programs will have to be stored and this will require a large amount of auxiliary storage.

(5) There is a disadvantage in having to program in basic assembler. The suggested trend in this thesis is towards generalization (for ease of programming) and away from machine oriented techniques. It is a backward step to have to include assembly routines with generalized routines. However in the long view the picture isn't so bleak. The inclusion of an assembly routine into RPG at least makes

the handling of the time sequence file possible. As experience grows this assembly routine will quite likely be generalized, firstly in a specific application and then as an industry wide programming system (see appendix VI).

In this chapter comment has been passed on the time sequence file and internal workings of RPG. Suggestions have been made on a general philosophy for using the technique. Impressions have been given on the actual application of the technique to large scale use. It remains only to express some conclusions.

CHAPTER V

CONCLUSIONS

This thesis has been a study of the computing concept of generalized programming and was divided into two parts.

Firstly it was a descriptive study of the concept of generalized data retrieval. As a particular example the System /360 Report Program Generator was presented. It was concluded that generalized data retrieval is applicable mainly to the one time management reporting requests. The concept is not very well suited for daily production reports since difficult report requirements cannot be programmed with the degree of refinement in generalized routines today. However the suggestion made was that as management's knowledge of computer capabilities develops they would request more one time reports in preference to the regular reports. Thus generalized data retrieval and RPG are programming systems for the near future.

Secondly, this thesis was a detailed study of a possible application for generalized data retrieval. A new idea in disk file organization, the time sequence file, has been suggested by B. A. Hodson and studied by Tiwari (21). An attempt was made in this thesis to use RPG as a programming base for retrieval of data from this file. To a limited extent RPG with modifications proved capable of retrieving data from this time sequence file. It appeared that on a small to medium scale application, in which the file remained fairly simple, RPG could retrieve data from it efficiently and effectively, with a minimum of

programming effort. The complete technique in detail for handling retrieval problems from the time sequence file by RPG has been explained in this thesis. It remains for further study to determine the application of these two concepts on a large scale (see appendix VI).

With close supervision over the three phases of file organization, file maintenance and data retrieval the time sequence file could prove very effective in certain applications. These would be of the variety in which data was constantly accumulated in the file and displayed for periodic inquiry. The data retrieval phase could also be solved on a large scale by RPG if close supervision is maintained. Similarities in retrieval requests must be recognized to keep the number of stored programs to a minimum. Novice programmers should not be given the retrieval job simply because RPG is oriented towards them. They would introduce unecessary overlapping and complications.

In summary, RPG as a generalized data retrieval system is capable of the retrieval job in certain applications. It has its limitations but can be very effective where these limitations are realized.

APPENDIX I

This appendix contains the following three program listings:

1. A program developed to create the time sequence disk files required to test the retrieval programs in the four examples. This program was written in RPG. It reads cards and formats them into records of 400 characters each and puts these on the disk (with standard labels) and an entry in the data set catalog. The program itself was catalogued on the disk and called whenever a test data set was required to be created for subsequent access by a retrieval program.
2. A listing of the example I computer decks divided into four parts:
   - i test data which is formatted into disk records - L1.
   - ii the RPG program written to retrieve this data - L2.
   - iii the assembler subroutine used to assist the retrieval of this data - L3.
   - iv the resulting report produced by the retrieval program-L4. In subsequent appendices the listings shown for example II, III and IV follow the same form as above.
3. A listing of the control card sequence for creating the data file, compiling RPG and assembler S/R, link edit and execution, is also presented in this appendix.

** L 1 ** GDR FILE CREATE RPG PROGRAM

```
00000H                                                                  CREATE
01010FCARDIN  IPEAF  80  80            READ40 SYSIPT                    CREATE
01020FMEDRE   O    F 400 400           DISK11 SYS007S                   CREATE
01030FRECOUT  O    F 132 132     OF    PRINTERSYSLST                    CREATE
02010ICARDIN  AA  01  80 D1                                             CREATE
02030I                                     1  80 REC1                   CREATE
02040I        BB  02  80 D2                                             CREATE
02050I                                     1  80 REC2                   CREATE
02060I        CC  03  80 D3                                             CREATE
02070I                                     1  80 REC3                   CREATE
02080I        DD  04  80 D4                                             CREATE
02090I                                     1  80 REC4                   CREATE
02100I        EE  05  80 D5                                             CREATE
02110I                                     1  80 REC5                   CREATE
03010OMEDRE   D        05                                               CREATE
030200                           REC1      80                          CREATE
030300                           REC2     160                          CREATE
030400                           REC3     240                          CREATE
030500                           REC4     320                          CREATE
030600                           REC5     400                          CREATE
030700RECOUT  H  301    OF                                              CREATE
030800        OR        1P                                              CREATE
030900                                  70 @TIME SEQUENCE DISK FILE@     CREATE
031000          D  2    01                                              CREATE
031100                           REC1     105                          CREATE
031200          D  2    02                                              CREATE
031300                           REC2     105                          CREATE
031400          D  2    03                                              CREATE
031500                           REC3     105                          CREATE
031600          D  2    04                                              CREATE
031700                           REC4     105                          CREATE
031800          D  2    05                                              CREATE
031900                           REC5     105                          CREATE
```

```
          ** L 1 **          GDR  EXAMPLE 1  TEST  DATA

00001M01     02008003     04    J B MITCHEL         39 RIVER ROAD                1
00001020160          4795246                                                      2
00001020240          6543210                                                      3
00001020320          2000101                                                      4
0000102              0034765                                                      5
00002M01     02008003     04    R TOMPSON           656 NORTH DRIVE              1
00002020160          4795246                                                      2
00002020240          6543210                                                      3
00002020320          2000101                                                      4
0000202              0039287                                                      5
00003M01     02008003     04    L SMALLMOUTH        17 ELDER AVENUE              1
00003020160          4795246                                                      2
00003020240          6543210                                                      3
00003020320          2000101                                                      4
0000302              0046583                                                      5
00004M01     02008003     04    ZEKE HANDLE         99  BUXTON ROAD              1
00004020160          4795246                                                      2
00004020240          6543210                                                      3
00004020320          2000101                                                      4
0000402              0092162                                                      5
00005M01     02008003     04    ARNOLD GLERK        69 RUE MARIE                 1
00005020160          4795246                                                      2
00005020240          6543210                                                      3
00005020320          2000101                                                      4
0000502              0069387                                                      5
```

** L 2 ** GDR EXAMPLE 1 RPG PROGRAM

```
00000H                                                                  EX1
01010FMEDRE   IPEAF 400 400          DISK11 SYS007S                     EX1
01020FAREPORT O   F 132 132     OF   PRINTERSYSLST                      EX1
02010IMEDRE   AA  01   6 CM                                            EX1
02020I                                    1   5 KEY                     EX1
02030I                                    7  30 XYTAB                   EX1
02040I                                   31  48 NAME                    EX1
02050I                                   49  64 ADDR                    EX1
02060I                                    1 200 REC                     EX1
02070I                                  201 400 REC2                    EX1
03010C          LOOP     TAG                                            EX1
03020C                   SETOF                    0302                  EX1
03030C                   EXIT GTSLR1                                    EX1
03040C                   RLABL          REC                             EX1
03070C                   RLABL          IN02                            EX1
03080C                   RLABL          IN03                            EX1
03090C                   ULABL          A      72                       EX1
03100C   02      A       COMP 11654.23            070607                EX1
03110C   02 07           GOTO LOOP                                      EX1
03120C   02 06   SUM     ADD  A         SUM    102                      EX1
040100AREPORT H  201     OF                                             EX1
040200         OR        1P                                             EX1
040300                                  50 @GDR TEST PROGRAM NO. 1  @   EX1
040400                                  75 @SEARCH  FOR  ONE  A PER @   EX1
040500                                  95 @LOGICAL RECORD@             EX1
040600          H  1     OF                                             EX1
040700         OR        1P                                             EX1
040800                                  10 @KEY@                        EX1
040900                                  40 @NAME@                       EX1
041000                                  70 @ADDRESS@                    EX1
041100                                 100 @FIELD A@                    EX1
041200          D  1     01                                             EX1
041300                           KEY    10                              EX1
041400                           NAME   50                              EX1
041500                           ADDR   75                              EX1
041600                   02      A     100 @    0.  @                  EX1
041650          D  1     01 03                                          EX1
041660                                  30 @VALIDITY ERROR@             EX1
041700          T 2      LR                                             EX1
041800                                  30 @SUMMARY OF FIELD A@         EX1
041900                           SUM    50 @       0.  @                EX1
```

```
** L 3 **          GDR  EXAMPLE 1  ASSEMBLER  SUBROUTINE

RPGSUB1  START 0
         ENTRY GTSLR1,A        SYMBOLS DEFINED HERE ARE USED IN RPG
         EXTRN REC,IN02,IN03   SYMBOLS DEFINED IN RPG ARE USED HERE
GTSLR1   BALR  15,0
         USING *,15
         B     BEGIN
SAVREG   DS    16F
ZERO     DC    C@    @
A        DC    PL4@0@
PACKED   DS    D
BEGIN    STM   0,15,SAVREG              STORE REGISTERS
         LM    2,4,#A%IN02,IN03,REC¤
         USING RECORD,4
         CLC   XYTAB+8%4¤,ZERO         IS SLR 02 PRESENT OR NOT
         BE    RETURN          NO SLR 02 TO ACCESS
MOVE     LR    5,4                     REG 5 NOW HAS ADDRESS OF REC
         PACK  PACKED,XYTAB+8%4¤
         CVB   6,PACKED
         AR    5,6                     REG 5 NOW HAS SLR 02 ADDRESS
         CLC   0%5,5¤,KEY              CHECK SLR KEY CODE
         BNE   ERR
         CLC   5%2,5¤,#C@02@           CHECK SLR ID  CODE
         BNE   ERR
         PACK  A,19%7,5¤
         MVC   XYTAB+8%4¤,7%5¤         UPDATE XYTAB WITH NEXT SLR
         MVI   0%2¤,X@F0@                    ADDRESS
         B     RETURN
ERR      MVI   0%3¤,X@F0@
RETURN   LM    0,15,SAVREG             RETURN REGISTER VALUES
         BR    14                      BRANCH TO RPG
RECORD   DSECT                  DUMMY CONTROL SECTION FOR RPG
KEY      DS    CL5                      VARIABLE NAMES
         DS    CL1                      THAT WILL BE
XYTAB    DS    CL24                     USED IN THIS S/R
         END
```

GDR TEST PROGRAM NO. 1 SEARCH FOR ONE A PE

LOGICAL RECORD

KEY	NAME	ADDRESS	FIELD A
00001	J B MITCHEL	39 RIVER ROAD	347.65
00002	R TOMPSON	656 NORTH DRIV	392.87
00003	L SMALLMOUTH	17 ELDER AVENU	465.83
00004	ZEKE HANDLE	99 BUXTON ROA	921.62
00005	ARNOLD GLERK	69 RUE MARIE	693.87

SUMMARY OF FIELD A 2821.84

IBM System/360 Assembler Coding Form

PROGRAM: JOB CONTROL CARDS FOR D.O.S. COMPILE AND
PROGRAMMER: EXECUTE RPG PLUS ASSEMBLER S/R
DATE:
PUNCHING INSTRUCTIONS:
GRAPHIC
PUNCH
PAGE OF
CARD ELECTRO NUMBER

STATEMENT

Name (1–8)	Operation (10–14)	Operand (16–20, 25, 30, 35, 40)	Comments (50, 55, 60, 65, 71)	Identification-Sequence (73–80)
// JOB	TEST	R KELTIE		
// ASSGN	SYS007,X'190'			
// VOL	SYS007,MEDRE			
// DLAB	'RPG TIME SEQUENCE DISK FILE		1111111',	C
		0001,66100,66100,'	',SD	
// XTENT	1,000,0001800000,0001800000,'111111'		,SYS007	
// EXEC	CREATE			
	test data deck			
/*				
// OPTION	LINK,LIST			
// EXEC	RPG			
	rpg source deck			
/*				
// OPTION	DUMP			
// EXEC	ASSEMBLY			
	assembler subroutine source deck			
/*				
	ENTRY	rpg program name		
// EXEC	LNKEDT			
// EXEC				
/&				

APPENDIX II

** L 1 ** GDR EXAMPLE 2 TEST DATA

```
00001M01    02006403    04   CHARLIE ABBOTT    737 UNDER ROAD    00001020096   1
   5712634      00001020128        24763        00001020160        3333377     2
00001020192        6732810     00001020224        1111199      00001020254   3
   4932876   00001020304        6739521                         00001020336   4
   6439518      00001020368        9996664        0000102          2674535     5
00002M01    02006403    04   JEAN VAL JEAN     LOWER PARIS ROAD  00002020096   1
   5712634      00002020128        65289        00002020160        3333377     2
00002020192        6732810     00002020224        1111199      00002020254   3
   4932876   00002020304        6739521                         00002020336   4
   6439518      00002020368        9996664        0000202          2674535     5
00003M01    02006403    04   HENRY BEDFORD     AVENUE OF ELMS    00003020096   1
   5712634      00003020128        91263        00003020160        3333377     2
00003020192        6732810     00003020224        1111199      00003020254   3
   4932876   00003020304        6739521                         00003020336   4
   6439518      00003020368        9996664        0000302          2674535     5
00004M01    02006403    04   SIMON P FRASER    63 CANOE BLVD     00004020096   1
   5712634      00004020128        44421        00004020160        3333377     2
00004020192        6732810     00004020224        1111199      00004020254   3
   4932876   00004020304        6739521                         00004020336   4
   6439518      00004020368        9996664        0000402          2674535     5
00005M01    02006403    04   RICHARD STEIN     NEW ORLEANS       00005020096   1
   5712634      00005020128        20874        00005070160        3333377     2
00005020192        6732810     00005020224        1111199      00005020254   3
   4932876   00005020304        6739521                         00005020336   4
   6439518      00005020368        9996664        0000502          2674535     5
```

** L 2 ** GDR EXAMPLE 2 RPG PROGRAM

```
00000H                                                                  EX2
01010FMEDRE   IPEAF 400 400          DISK11 SYS007S                     EX2
01020FAAREPORTO   F 132 132     OF   PRINTERSYSLST                      EX2
02010IMEDRE   AA  01   6 CM                                            EX2
02020I                                    1   5 KEY                     EX2
02030I                                    7  30 XYTAB                   EX2
02040I                                   31  48 NAME                    EX2
02050I                                   49  64 ADDR                    EX2
02060I                                    1 200 REC                     EX2
02070I                                  201 400 REC2                    EX2
03010C                SETOF                      0213                   EX2
03020C                SETOF                      141516                 EX2
03030C                EXIT GTSLR2                                       EX2
03040C                RLABL          REC                                EX2
03070C                RLABL          IN02          VALIDITY  CHECK      EX2
03080C                RLABL          IN13                               EX2
03090C                RLABL          IN14                               EX2
03100C                RLABL          IN15                               EX2
03110C                RLABL          IN16                               EX2
03120C                ULABL          A1      72                         EX2
03130C                ULABL          A2      72                         EX2
03140C                ULABL          A3      72                         EX2
03150C                ULABL          A4      72                         EX2
03160C                ULABL          A5      72                         EX2
03170C                ULABL          A6      72                         EX2
03180C                ULABL          A7      72                         EX2
03190C                ULABL          A8      72                         EX2
03200C                ULABL          A9      72                         EX2
04010C                ULABL          A10     72                         EX2
04020C                ULABL          A11     72                         EX2
04030C                ULABL          A12     72                         EX2
050100AAREPORTH  201   OF                                                EX2
050200        OR      1P                                                EX2
050300                              50 @GDR TEST PROGRAM NO. 2  @         EX2
050400                              75 @SEARCH FOR ALL A IN  @            EX2
050500                              89 @LOGICAL RECORD@                   EX2
050600         H  1   OF                                                 EX2
050700        OR      1P                                                 EX2
050800                              10 @KEY@                              EX2
050900                              40 @NAME@                             EX2
051000                              70 @ADDRESS@                          EX2
051100                             117 @          A   VALUES     @        EX2
051300         D 11   01                                                 EX2
051400                      KEY     10                                   EX2
051500                      NAME    50                                   EX2
051600                      ADDR    75                                   EX2
051700         D  1   13                                                 EX2
051800                      A1     103 @    0.  @                        EX2
051900                      A2     114 @    0.  @                        EX2
052000                      A3     125 @    0.  @                        EX2
060100         D  1   14                                                 EX2
060200                      A4     103 @    0.  @                        EX2
060300                      A5     114 @    0.  @                        EX2
060400                      A6     125 @    0.  @                        EX2
060500         D  1   15                                                 EX2
060600                      A7     103 @    0.  @                        EX2
060700                      A8     114 @    0.  @                        EX2
```

```
** L 2 **                GDR  EXAMPLE 2  RPG    PROGRAM

060800                        A9       125 @    0.  @                  EX2
060900          D  1     16                                             EX2
061000                        A10      103 @    0.  @                  EX2
061100                        A11      114 @    0.  @                  EX2
061200                        A12      125 @    0.  @                  EX2
061300          D  1     02                                             EX2
061400                                  30 @VALIDITY ERROR@             EX2
```

```
                ** L 3 **          GDR  EXAMPLE 2  ASSEMBLER  SUBROUTINE

RPGSUB2  START 0
         ENTRY GTSLR2,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12
         EXTRN REC,IN02,IN13
GTSLR2   BALR  15,0
         USING *,15
         B     BEGIN
SAVREG   DS    16F
ZERO     DC    C@    @
A1       DC    PL4@0@                    CONSTANT AREA FILLED IN S/R
A2       DC    PL4@0@
A3       DC    PL4@0@
A4       DC    PL4@0@
A5       DC    PL4@0@
A6       DC    PL4@0@
A7       DC    PL4@0@
A8       DC    PL4@0@
A9       DC    PL4@0@
A10      DC    PL4@0@
A11      DC    PL4@0@
A12      DC    PL4@0@
PACKED   DS    D
BEGIN    STM   0,15,SAVREG
         L     4,#A%REC¤
         USING RECORD,4
         L     2,#A%IN02¤
         L     7,#A%IN13¤
         LA    8,A1
         MVI   0%8¤,X@00@                ZERO OUT CONSTANT AREA
         MVC   1%47,8¤,0%8¤
         L     3,#F@0@                   COUNT NUMBER OF CONSTANTS
COMP     CLC   XYTAB+8%4¤,ZERO                 FILLED
         BE    RETURN
         LR    5,4                       REG5 HAS REC ADDRESS NOW
         PACK  PACKED,XYTAB+8%4¤
         CVB   6,PACKED
         AR    5,6                       REG5 HAS SLR ADDRESS NOW
         CLC   0%5,5¤,KEY
         BNE   VERR
         CLC   5%2,5¤,#C@02@
         BNE   VERR
         C     3,#F@03@
         BNE   MOVE
         A     7,#F@01@      EVERY 3 CONSTANTS FILLED SETS 1 INDICATOR
         L     3,#F@0@
MOVE     A     3,#F@01@
         MVI   0%7¤,X@F0@
         PACK  0%4,8¤,19%7,5¤
         MVC   XYTAB+8%4¤,7%5¤
         A     8,#F@04@              UPDATE ADDRESS OF LAST UNFILLED
         B     COMP                                CONSTANT
VERR     MVI   0%2¤,X@F0@            SET ERROR INDICATOR
RETURN   LM    0,15,SAVREG
         BR    14
RECORD   DSECT                       DUMMY CONTROL SECTION FOR RPG
KEY      DS    CL5                         VARIABLE NAMES
         DS    CL1                         THAT WILL BE
XYTAB    DS    CL24                      USED IN THIS S/R
```

```
** L 3 **          GDR  EXAMPLE 2  ASSEMBLER  SUBROUTINE

      END
```

GDR TEST PROGRAM NO. 2 SEARCH FOR ALL A IN LOGICAL RECORD

KEY	NAME	ADDRESS	A VALUES		
00001	CHARLIE ABBOTT	737 UNDER ROAD	57126.34	247.63	33333.77
			67328.10	11111.99	49328.76
			67395.21	64395.18	99966.64
			26745.35	.00	.00
00002	JEAN VAL JEAN	LOWER PARIS ROAD	57126.34	652.89	33333.77
			67328.10	11111.99	49328.76
			67395.21	64395.18	99966.64
			26745.35	.00	.00
00003	HENRY BEDFORD	AVENUE OF ELMS	57126.34	912.63	33333.77
			67328.10	11111.99	49328.76
			67395.21	64395.18	99966.64
			26745.35	.00	.00
00004	SIMON P FRASER	63 CANOE BLVD	57126.34	444.21	33333.77
			67328.10	11111.99	49328.76
			67395.21	64395.18	99966.64
			26745.35	.00	.00
00005	RICHARD STEIN	NEW ORLEANS	57126.34	208.74	.00
	VALIDITY ERROR				

APPENDIX III

** L 1 ** GDR EXAMPLE 3 TEST DATA

```
00001M01    02    03008004    HENRY THOREAU     CONCORD MASS.                    1
000010302401F YOU HAVE BUILT CASTLES IN THE AIR,YOUR WORK NEED NOT BE LOST,01602
0000103     THAT IS WHERE THEY SHOULD BE. NOW PUT THE FOUNDATIONS UNDER THEM     3
0000103     IF A MAN DOES NOT KEEP PACE WITH HIS COMPANIONS, PERHAPS IT IS  03204
0000103     BECAUSE HE HEARS A DIFFERENT DRUMMER.                            5
00002M01    02    03008004    ROBERT FROST      NEW ENGLAND                    1
000020302401HE WAY A CROW    SHOOK DOWN ON ME                              01602
0000203     THE DUST OF SNOW    FROM A HEMLOCK TREE                          3
0000203     HAS GIVEN MY HEART    A CHANGE OF MOOD                       03204
0000203     AND SAVED SOME PART    OF A DAY I HAD RUED                       5
00003M01    02    03008004    BERTRAND RUSSEL   LONDON ENGLAND                 1
0000303     I THINK THAT THERE IS FAR TOO MUCH WORK DONE IN THE WORLD,    01602
0000303     THAT IMMENSE HARM IS CAUSED BY THE BELIEF THAT WORK IS VIRTUOUS,02403
0000303      AND THAT WHAT NEEDS TO BE PREACHED IN MODERN INDUSTRIAL SOCIETY03204
0000303     IS QUITE DIFFERENT FROM WHAT ALWAYS HAS BEEN PREACHED.           5
00004M01    02    03008004    GEORGE ORWELL     ANIMAL FARM                    1
0000403     THE CREATURES OUTSIDE LOOKED FROM PIG TO MAN,AND FROM MAN TO PIG01602
0000403     AND FROM PIG TO MAN  AGAIN,  BUT ALREADY IT WAS IMPOSSIBLE    02403
0000403     TO SAY WHICH WAS WHICH.                                          4
                                                                             5
00005M01    02    03008004    LUDWIG  BEETHOVEN BONN  GERMANY                 1
000050303202HE IS NOT AN ARTISAN BUT AN ARTIST, A CHAMPION OF HUMAN FREEDOM,01602
0000503     A BROODING, OVERBEARING PERSONALITY, WHOSE PRIVATE LEGEND IT IS 02403
0000503     TEMPERING TO FORCE INTO A PARALLEL WITH HIS ARTISTIC  ONE.        4
0000503     A POPULAR HISTORY OF MUSIC BY CARTER  HARMAN                     5
```

** L 2 ** GDR EXAMPLE 3 RPG PROGRAM

```
00000H                                                                  EX3
01010FMEDRE   IPEAF 400 400            DISK11 SYS007S                   EX3
01020FALPHARPTO   F 132 132     OF    PRINTERSYSLST                    EX3
02010IMEDRE   AA  01   6 CM                                            EX3
02020I                                      1   5 KEY                   EX3
02030I                                      7  30 XYTAB                 EX3
02040I                                     31  48 NAME                  EX3
02050I                                     49  64 ADDR                  EX3
02060I                                      1 200 REC                   EX3
02070I                                    201 400 REC2                  EX3
03010C                  SETOF                     021314                EX3
03020C                  EXIT GTSLR3                                     EX3
03030C                  RLABL      REC                                  EX3
03060C                  RLABL      IN02                                 EX3
03070C                  RLABL      IN13                                 EX3
03080C                  RLABL      IN14                                 EX3
03090C                  ULABL      ALPHA1 64                            EX3
03100C                  ULABL      ALPHA2 64                            EX3
03110C                  ULABL      ALPHA3 64                            EX3
03120C                  ULABL      ALPHA4 64                            EX3
03130C                  ULABL      ALPHA5 64                            EX3
03140C                  ULABL      ALPHA6 64                            EX3
040100ALPHARPTH  201    OF                                              EX3
040200       OR         1P                                              EX3
040300                                 50 @GDR TEST PROGRAM NO. 3  @    EX3
040400                                 75 @SEARCH FOR ALPHA TEXT@       EX3
040500                                 94 @ IN LOGICAL RECORD@          EX3
040600         H  1     OF                                              EX3
040700       OR         1P                                              EX3
040800                                  7 @KEY@                        EX3
040900                                 20 @NAME@                       EX3
041000                                 45 @ADDRESS@                    EX3
041100                                 90 @TEXT@                       EX3
041200         D 11     01                                              EX3
041300                        KEY       7                              EX3
041400                        NAME     30                              EX3
041500                        ADDR     50                              EX3
041600         D  1     13                                              EX3
041700                        ALPHA1   120                             EX3
041800         D  1     13                                              EX3
041900                        ALPHA2   120                             EX3
050100         D  1     13                                              EX3
050200                        ALPHA3   120                             EX3
050300         D  1     14                                              EX3
050400                        ALPHA4   120                             EX3
050500         D  1     14                                              EX3
050600                        ALPHA5   120                             EX3
050700         D  1     14                                              EX3
050800                        ALPHA6   120                             EX3
050900         D  1     02                                              EX3
051000                                 30 @VALIDITY ERROR@              EX3
```

```
                ** L 3 **          GDR  EXAMPLE 3  ASSEMBLER  SUBROUTINE

RPGSU83  START 0
         ENTRY GTSLR3,ALPHA1,ALPHA2,ALPHA3,ALPHA4,ALPHA5,ALPHA6
         EXTRN REC,IN02,IN13
GTSLR3   BALR  15,0
         USING *,15
         B     BEGIN
SAVREG   DS    16F
ZERO     DC    C@    @
ALPHA1   DC    CL64@ @                    ALPHABETIC STORE AREA FOR SLR
ALPHA2   DC    CL64@ @                    RETURN TO RPG
ALPHA3   DC    CL64@ @
ALPHA4   DC    CL64@ @
ALPHA5   DC    CL64@ @
ALPHA6   DC    CL64@ @
PACKED   DS    D
ACHAIN   DC    C@    @                    CHAIN ADDRESS POINTER IN SLR
BEGIN    STM   0,15,SAVREG
         L     4,#A%REC¤
         USING RECORD,4
         L     2,#A%IN02¤
         L     7,#A%IN13¤
         LA    8,ALPHA1
         MVI   0%8¤,X@40@
         MVC   1%200,8¤,0%8¤              ZERO OUT STORE AREA
         MVC   201%183,8¤,200%8¤
         L     3,#F@0@
COMP     CLC   XYTAB+14%4¤,ZERO
         BE    ENDTXT                     BRANCH TO CHECK FOR MORE SLR
         LR    5,4                                DATA
         PACK  PACKED,XYTAB+14%4¤
         CVB   6,PACKED
         AR    5,6
         CLC   0%5,5¤,KEY
         BNE   VERR
         CLC   5%2,5¤,#C@03@
         BNE   VERR
         C     3,#F@03@
         BNE   MOVE
         A     7,#F@01@                   REG7 HAS INDICATOR ADDRESS
         L     3,#F@0@                    REG3 HAS SLR COUNT PER
MOVE     A     3,#F@01@                           INDICATOR
         MVI   0%7¤,X@F0@
         MVC   0%64,8¤,11%5¤
         MVC   XYTAB+14%4¤,75%5¤          BCHAIN POINTS TO MORE OF SAME
         CLC   7%4,5¤,ZERO                         MESSAGE
         BE    *+10
         MVC   ACHAIN%4¤,7%5¤
         A     8,#F@64@
         B     COMP
ENDTXT   CLC   ACHAIN%4¤,ZERO
         BE    RETURN
         MVC   XYTAB+14%4¤,ACHAIN         ACHAIN POINTS TO NEW MESSAGE
         MVC   ACHAIN%4¤,ZERO
         B     COMP
VERR     MVI   0%2¤,X@F0@
RETURN   LM    0,15,SAVREG
         BR    14
```

```
** L 3 **          GDR  EXAMPLE 3  ASSEMBLER  SUBROUTINE

RECORD   DSECT                    DUMMY CONTROL SECTION FOR RPG
KEY      DS      CL5                      VARIABLE NAMES
         DS      CL1                      THAT WILL BE
XYTAB    DS      CL24                   USED IN THIS S/R
         END
```

GDR TEST PROGRAM NO. 3 SEARCH FOR ALPHA TEXT IN LOGICAL RECORD

KEY	NAME	ADDRESS	TEXT
00001	HENRY THOREAU	CONCORD MASS.	IF YOU HAVE BUILT CASTLES IN THE AIR,YOUR WORK NEED NOT BE LOST, THAT IS WHERE THEY SHOULD BE. NOW PUT THE FOUNDATIONS UNDER THEM IF A MAN DOES NOT KEEP PACE WITH HIS COMPANIONS, PERHAPS IT IS BECAUSE HE HEARS A DIFFERENT DRUMMER.
00002	ROBERT FROST	NEW ENGLAND	THE WAY A CROW SHOOK DOWN ON ME THE DUST OF SNOW FROM A HEMLOCK TREE HAS GIVEN MY HEART A CHANGE OF MOOD AND SAVED SOME PART OF A DAY I HAD RUED
00003	BERTRAND RUSSEL	LONDON ENGLAND	I THINK THAT THERE IS FAR TOO MUCH WORK DONE IN THE WORLD, THAT IMMENSE HARM IS CAUSED BY THE BELIEF THAT WORK IS VIRTUOUS, AND THAT WHAT NEEDS TO BE PREACHED IN MODERN INDUSTRIAL SOCIETY IS QUITE DIFFERENT FROM WHAT ALWAYS HAS BEEN PREACHED.
00004	GEORGE ORWELL	ANIMAL FARM	THE CREATURES OUTSIDE LOOKED FROM PIG TO MAN,AND FROM MAN TO PIG AND FROM PIG TO MAN AGAIN, BUT ALREADY IT WAS IMPOSSIBLE TO SAY WHICH WAS WHICH.
00005	LUDWIG BEETHOVEN	BONN GERMANY	HE IS NOT AN ARTISAN BUT AN ARTIST, A CHAMPION OF HUMAN FREEDOM, A BROODING, OVERBEARING PERSONALITY, WHOSE PRIVATE LEGEND IT IS TEMPERING TO FORCE INTO A PARALLEL WITH HIS ARTISTIC ONE. A POPULAR HISTORY OF MUSIC BY CARTER HARMAN

APPENDIX IV

** L 1 ** GDR EXAMPLE 4 TEST DATA

```
00001M01    02    03    040080BARRY HARTRY      100 DAFOE ROAD                  1
0000104016077777666633345297HARVEY47123                                         2
0000104024033333555544427644HENDEL69702                                         3
0000104028011111999922287164HENSON45678 0000104032099999333311127654HEROLD453454
0000104    55555222299967921HELPER98124                                         5
00002M01    02    03    040080ROBERT CALLUM     440 CENTENNIAL                  1
0000204016077777666633345297CAHOON47123                                         2
0000204024033333555544427644JERZAK69702                                         3
0000204028011111999922287164MATISZ45678 0000204032099999333311127654OKEEFE453454
0000204    55555222299967921SIMEON98124                                         5
00003M01    02    03    040080BILL BUCKWOLD     357 OXFORD                      1
0000304016077777666633345297BURTON47123                                         2
0000304024033333555544427644LEPPKY69702                                         3
0000304028011111999922287164MARTIN45678 0000304032099999333311127654OLESEN453454
0000304    55555222299967921RONALD98124                                         5
00004M01    02    03    040080LEO JOHNSON       299 WAVERLY                     1
0000404016077777666633345297BUOCK 47123                                         2
0000404024033333555544427644LEIBEL69702                                         3
0000404028011111999922287164MASLEN45678 0000404032099999333311127654NORTON453454
0000404    55555222299967921LAUREN98124                                         5
00005M01    02    03    040080SANNE JENSEN      STE AGATHE                      1
0000504016077777666633345297BUNKA 47123                                         2
0000504024033333555544427644LECKEL69702                                         3
0000504028011111999922287164MATSUO45678 0000504032099999333311127654NISBET453454
0000504036055555222299967921THOMAS98124 0000504    444446666777275 61DONALD611125
```

** L 2 ** GDR EXAMPLE 4 RPG PROGRAM

```
00000H                                                                  EX4
01010FMEDRE   IPEAF 400 400                DISK11 SYS007S               EX4
01020FREPORT040   F 132 132      OF        PRINTERSYSLST                EX4
02010IMEDRE   AA  01   6 CM                                            EX4
02020I                                       1   5 KEY                  EX4
02030I                                       7  30 XYTAB                EX4
02040I                                      31  48 NAME                 EX4
02050I                                      49  64 ADDR                 EX4
02060I                                       1 200 REC                  EX4
02070I                                     201 400 REC2                 EX4
03010C                  SETOF                      020315               EX4
03020C                  SETOF                      161718               EX4
03030C                  SETOF                      19                   EX4
03040C                  EXIT GTSLR4                                     EX4
03050C                  RLABL          REC                              EX4
03080C                  RLABL          IN02                             EX4
03090C                  RLABL          IN03                             EX4
03100C                  RLABL          IN15                             EX4
03110C                  RLABL          IN16                             EX4
03120C                  RLABL          IN17                             EX4
03130C                  RLABL          IN18                             EX4
03140C                  RLABL          IN19                             EX4
03150C                  ULABL          MASK1  44                        EX4
03160C                  ULABL          MASK2  44                        EX4
03170C                  ULABL          MASK3  44                        EX4
03180C                  ULABL          MASK4  44                        EX4
03190C                  ULABL          MASK5  44                        EX4
03200C                  ULABL          OFCNT   30                       EX4
04010OREPORT04H  201     OF                                             EX4
040200        OR        1P                                             EX4
040300                                  50 @GDR TEST PROGRAM NO. 4  @   EX4
040400                                  75 @SEARCH FOR ALL 04 SLR@      EX4
040500                                  94 @ IN LOGICAL RECORD@         EX4
040600         H  1     OF                                             EX4
040700        OR        1P                                             EX4
040800                                   7 @KEY@                       EX4
040900                                  20 @NAME@                      EX4
041000                                  45 @ADDRESS@                   EX4
041100                                  80 @04 ID SLR@                 EX4
041200         D 11     01                                             EX4
041300                        KEY       7                             EX4
041400                        NAME     30                             EX4
041500                        ADDR     50                             EX4
041600         D  1     15                                             EX4
041700                        MASK1    100                            EX4
041800         D  1     16                                             EX4
041900                        MASK2    100                            EX4
050100         D  1     17                                             EX4
050200                        MASK3    100                            EX4
050300         D  1     18                                             EX4
050400                        MASK4    100                            EX4
050800         D  1     19                                             EX4
050900                        MASK5    100                            EX4
051000         D  1     02                                             EX4
051100                                  30 @VALIDITY ERROR@            EX4
051200         D  1     03                                             EX4
051300                                  40 @OVERFLOW BY @              EX4
```

** L 2 ** GDR EXAMPLE 4 RPG PROGRAM

```
051400                    OFCNT     50 @   @                    EX4
051500                              60 @RECORDS@                 EX4
```

** L 3 ** GDR EXAMPLE 4 ASSEMBLER SUBROUTINE

```
RPGSUB4  START 0
         ENTRY GTSLR4,MASK1,MASK2,MASK3,MASK4,MASK5,OFCNT
         EXTRN REC,IN02,IN03,IN15
GTSLR4   BALR  15,0
         USING *,15
         B     BEGIN
SAVREG   DS    16F
ZERO     DC    C@    @
MASK1    DC    CL44@   .   .   .   .   @
MASK2    DC    CL44@   .   .   .   .   @
MASK3    DC    CL44@   .   .   .   .   @
MASK4    DC    CL44@   .   .   .   .   @
MASK5    DC    CL44@   .   .   .   .   @
PACKED   DS    D
OFCNT    DC    PL2@0@
BEGIN    STM   0,15,SAVREG
         LM    2,4,#A%IN02,IN03,REC¤
         USING RECORD,4
         L     7,#A%IN15¤
         LA    8,MASK1
         SP    OFCNT,OFCNT              OVERFLOW COUNT
         L     9,#F@0@                  SLR RETRIEVED COUNT
COMP     CLC   XYTAB+20%4¤,ZERO
         BE    RETURN
         LR    5,4
         PACK  PACKED,XYTAB+20%4¤
         CVB   6,PACKED
         AR    5,6
         CLC   0%5,5¤,KEY
         BNE   VERR
         CLC   5%2,5¤,#C@04@
         BNE   VERR
         C     9,#F@05@
         BE    OFERR
MOVE     A     9,#F@01@                 COUNT NUMBER OF SLR RETRIEVED
         MVI   0%7¤,X@F0@
         A     7,#F@01@                 UPDATE INDICATOR ADDRESS
         MVC   XYTAB+20%4¤,7%5¤
         MVC   0%4,8¤,11%5¤             MOVE DATA FIELDS INTO MASK
         MVC   5%1,8¤,15%5¤
         MVC   9%4,8¤,16%5¤
         MVC   14%3,8¤,20%5¤
         MVC   20%4,8¤,23%5¤
         MVC   25%1,8¤,27%5¤
         MVC   29%6,8¤,28%5¤
         MVC   38%1,8¤,34%5¤
         MVC   40%4,8¤,35%5¤
         A     8,#F@44@
         B     COMP
OFERR    MVI   0%3¤,X@F0@               SET INDICATOR FOR RPG MESSAGE
         AP    OFCNT,#P@1@              COUNT OVERFLOW SLR
         MVC   XYTAB+20%4¤,7%5¤
         B     COMP
VERR     MVI   0%2¤,X@F0@
RETURN   LM    0,15,SAVREG
         BR    14
RECORD   DSECT                         DUMMY CONTROL SECTION FOR RPG
```

```
** L 3 **          GDR  EXAMPLE 4  ASSEMBLER  SUBROUTINE

KEY       DS    CL5                    VARIABLE NAMES
          DS    CL1                    THAT WILL BE
XYTAB     DS    CL24                   USED IN THIS S/R
          END
```

GDR TEST PROGRAM NO. 4 SEARCH FOR ALL 04 SLR IN LOGICAL RECORD

KEY	NAME	ADDRESS	04 ID SLR				
00001	BARRY HARTRY	100 DAFOE ROAD					
			7777.7	6666.333	4529.7	HARVEY	4.7123
			3333.3	5555.444	2764.4	HENDEL	6.9702
			1111.1	9999.222	8716.4	HENSON	4.5678
			9999.9	3333.111	2765.4	HEROLD	4.5345
			5555.5	2222.999	6792.1	HELPER	9.8124
00002	ROBERT CALLUM	440 CENTENNIAL					
			7777.7	6666.333	4529.7	CAHOON	4.7123
			3333.3	5555.444	2764.4	JERZAK	6.9702
			1111.1	9999.222	8716.4	MATISZ	4.5678
			9999.9	3333.111	2765.4	OKEEFE	4.5345
			5555.5	2222.999	6792.1	SIMEON	9.8124
00003	BILL BUCKWOLD	357 OXFORD					
			7777.7	6666.333	4529.7	BURTON	4.7123
			3333.3	5555.444	2764.4	LEPPKY	6.9702
			1111.1	9999.222	8716.4	MARTIN	4.5678
			9999.9	3333.111	2765.4	OLESEN	4.5345
			5555.5	2222.999	6792.1	RONALD	9.8124
00004	LEO JOHNSON	299 WAVERLY					
			7777.7	6666.333	4529.7	BUOCK	4.7123
			3333.3	5555.444	2764.4	LEIBEL	6.9702
			1111.1	9999.222	8716.4	MASLEN	4.5678
			9999.9	3333.111	2765.4	NORTON	4.5345
			5555.5	2222.999	6792.1	LAUREN	9.8124
00005	SANNE JENSEN	STE AGATHE					
			7777.7	6666.333	4529.7	BUNKA	4.7123
			3333.3	5555.444	2764.4	LECKEL	6.9702
			1111.1	9999.222	8716.4	MATSUO	4.5678
			9999.9	3333.111	2765.4	NISBET	4.5345
			5555.5	2222.999	6792.1	THOMAS	9.8124

OVERFLOW BY 1 RECORDS

APPENDIX V

A SUMMARY OF PROGRAMMING REQUIREMENTS

For programming retrieval problems with the time sequence file using the technique suggested in this thesis, the following list of points should be kept in mind:

(1) The retrieval technique suggested is a combination of the System /360, Report Program Generator and an Assembler Subroutine. Therefore a thorough knowledge is required of both languages (RPG and Basic Assembler). The assembler routine may require some input/output macros as will be explained later so a familiarity with these is also required. As reference material the following publications are suggested. (see the Bibliography for manual file and form numbers)

IBM /360 : Introduction

IBM /360 : Concept and Facilities

IBM /360 : Data Management

IBM /360 : Principle of Operation

IBM /360 : Assembler Language

IBM /360 : Report Program Generator

IBM /360 : System Control and Service Programs

IBM /360 : Control Program Services, section on Data Management Services.

(2) The time sequence file must exist in a document form and

should be set up according to the sub-record (SLR) concept using control fields as suggested in this thesis and by Tiwari (21).

(3) Information is first coded on the RPG report forms beginning with File Description Specifications.

(a) The time sequence file can have any name, but it must be identical to that given for it at the time of file creation. This name resides on the Volume Table of Contents on the Disk (see 12, manual 4, page 29).

(b) The time sequence file is indicated as follows:

Input file	I
Primary file	P
End of file	E
Ascending sequence	A
File Format	F

(c) The block length equals the record length which is the full record length, including undefined SLR area.

(d) The rest of the entries are variable and will depend upon the application. In the 4 examples a sequential disk file was used so columns 28-39 are blank.

(e) The DEVICE name and SYMBOLIC DEVICE must correspond to that disk on which the file resides.

(4) No File Extension or Line Counter Specifications will normally be required if all the file is to be searched. If only selected records are to be searched a table look

up is used. The File Extension Specifications describe this table. Line Counter Specifications are needed only if the report is to be stored temporarily and retrieved for later printing.

(5) Input Specifications are next:

(a) On the first line the file is described. It requires a resulting indicator field and record identification. This identification must appear in the MLR portion of the record. In the 4 examples each record was identified as M in position 6.

(b) On subsequent lines the record is described in detail. Each MLR field should be described. Its relative record positions are known. The x-y table should be described as one field as in the examples.

(c) The whole record must be broken down into sections of 256 characters (or less) each. In the examples two sections of 200 characters each were chosen.

(6) On the calculation specifications the following must appear:

(a) At the beginning indicators tested in the assembler subroutine should be SET OFF.

(b) Next and EXIT to the assembler subroutine (named) must occur (see 12, manual 1, pages 107 - 109).

(c) The macro RLABL must be used in the RPG program to give the subroutine the address of all variables on the input record. This includes all MLR fields to be used in the

assembler routine. It also includes the first section of the record defined on Input specs (REC). This procedure gives the assembler subroutine addresses of the record and fields it needs for searching.

(d) Any variables generated by the assembler routine and required for calculation and/or printing by the RPG program must appear in a ULABL statement. Their character lengths and decimal positions are specified in columns 49 - 52.

(e) The EXIT is encountered only once for each record read in by RPG from the disk. This means that all fields to be accessed by the assembly routine are found at once. Return to the RPG program means returning with all desired fields in the record. Each of these fields must have a different name and appear in a ULABL statement. Moreover, a maximum number of possible fields must be allotted for in the ULABL statements. Each field (or group of fields) found in the assembler routine has associated with it (them) an RPG indicator which is set on if this field is found. The indicator is then tested in the Output specifications. If it is on, then the field is printed, and if it isn't then the field isn't printed.

(7) Output Specifications are the final RPG requirements.

(a) Headings are coded the same for any report.

(b) The MLR information will normally appear for each record accessed. This includes such identification information

as name and address. It's printing is triggered by the indicator set on for the record in RESULTING INDICATOR on Input Specifications (columns 19 - 20).

(c) The printing of the SLR fields is triggered by indicators set in the assembler program. All possible variables must be coded for printing on the output specifications. At print time only those whose indicators are on will be printed. The indicator off means that the variable hasn't been filled from the record searched in the assembler program.

(d) Summary information may be printed at the end of the report.

(8) Attention now turns to the Assembler subroutine in which the following items must be included.

(a) The program can have any name which appears in the NAME field of a START instruction.

(b) The next instruction should be an ENTRY instruction whose OPERAND is the same name that appears in FACTOR 2. of the RPG EXIT statement. This name must also appear as the NAME of the BALR instruction which starts the subroutine.

(c) Variables coming from the RPG program must appear next in EXTRN per variable. In DOS all variables may appear in an EXTRN statement.

Note: only the first section of the record which has been subdivided into 200 character sections need appear in an EXTRN statement. An ENTRY defines the retrieved fields.

(d) The program begins with BALR and USING instructions. The NAME of the BALR is the FACTOR 2 field in the EXIT statement in RPG.

(e) Registers must be saved now and later restored when returning to RPG.

(f) The maximum number of fields that can be retrieved from a record must be defined in this program. They are defined as separately named variables. They are returned to zero before searching begins and are consecutively filled as slots when another field that meets searching requirements is found.

(g) Indicators from the RPG program are set at the same time as a slot is filled to indicate to the RPG program that the slot is available for calculation and/or printing. The addressing of these slots and indicators as each field in the record is located is illustrated in the four EXAMPLES. Turning on the indicator is accomplished as explained in the RPG manual (12, manual 1).

(h) - Numeric characters in the record appear in zoned decimal format (see 12, Assembler Language Manual, Appendix A).

- Alphabetic characters in the record appear in character format (see 12, Assembler Language Manual, Appendix A, page 103).

- Numeric fields returned to the RPG program must be converted to packed decimal (they are numeric if required for

calculation before printing).

- Alphabetic and numeric fields required for printing only can be returned to the RPG program just as they appear in the record.
- Numeric fields from the RPG program used in the assembler program appear in packed decimal format (use of these is rare).
- Indicators occupy one byte of core and are set on as explained in the RPG manual (12).

(i) Error indicators are also set if required by the assembler routine and tested at print time by the RPG program. They are of two types:

(1) Validity Error

- When a SLR has a KEY which doesn't correspond to that of the MLR.
- When an SLR identification code doesn't match that of the SLR to which it is chained.

This error occurs when data has been put in the record improperly (i.e. an error in the file maintenance program).

(2) Overflow Error

- When too few slots have been allotted in the program to access all the possible fields in a particular record. The amount of overflow should be programmed and written out on the RPG report. The program will

then have to be rewritten to accomodate the extra SLR's appearing in this particular record. This error will occur less and less often as knowledge of the environment of a particular SLR grows.

(9) There is the possibility of overflow of a different type. It has been explained in part II chapter I and is fully described by Tiwari (21). In order to accomodate an overflow of this type I/O macros must be included in the Assembler routine along with additional coding to determine when the overflow area has to be accessed. The I/O macros will depend upon the organization of the overflow area which is dependent upon the particular application. The accessing of this overflow area takes place at the beginning of the assembler routine. When the searching logic comes to the end of the record it immediately goes to the overflow record and searches it using the same logic. The assembler routine must therefore know:

(1) That an overflow record is available for this record.

(2) What its core address will be on access.

(3) The lengths of the main record and overflow record.

It must access the overflow record from disk when one is indicated in the main record. It must have the additional coding to sense the end of the main record and then search the overflow record. It must sense the end of the overflow record and stop searching.

(10) When the two programs have been written they are put together with Job Control Cards for compiling and execution. The input deck is described on pages 135-140 of the BOS RPG manual (12, manual 1). This is the System /360 DISK OPERATING SYSTEM control card sequence (see appendix I). Other operating systems will require modified versions of this control card sequence.

APPENDIX VI

SUGGESTED APPROACH FOR GENERALIZING THE ASSEMBLER SUBROUTINE

Implementation of RPG plus assembler subroutine programming on a large scale needs some additional comments. This appendix is intended to provide some suggestions which will assist in the establishment of full scale use of the above discussed data retrieval concept.

As experience grows with the use of the RPG - assembler S/R data retrieval technique various trends should become evident. The first trend will be towards the retrieval of all SLR's of one type for a certain account record (i.e. student record) upon random request. The person making the request will most often be interested in obtaining all the facts on a certain topic for an account (all chemistry exam results). He may be interested in only one SLR which meets his needs (the chemistry exam results over 70%), but he will want all information for an SLR. This is significant because it then allows the programming effort to be divided into jobs of producing one retrieval program for each SLR. Most requests for information from this SLR can be satisfied by this one program. It should produce the SLR as in example 4 with the option of retrieving only one copy depending upon some comparison condition as in example 1. So for each SLR designed for the file one retrieval program must be written and catalogued on disk for periodic access. It will describe the file, arithmetic processing, and report

format in RPG and will include an assembler subroutine to search for the SLR's. No generalization of the subroutine is possible under these circumstances. Each SLR will require separate RPG report requirements and separate assembler searching and slotting requirements. There is a similarity in form of each SLR program but it is not close enough to allow one routine to be written for all. It appears reasonable that routines such as example 4 could quite easily be written for all SLR's once a few are developed.

Generalization does, however, have its place in the RPG - assembler S/R data retrieval concept. RPG is a generalized routine. It creates a specific program from parameter card input. However, the assembler S/R is still specific. Indeed, for the complete SLR retrieval mentioned above, the subroutine cannot be generalized. Each SLR format is quite different from the next.

A class of retrieval problems not covered by complete SLR retrieval still exists and it is in this area that the subroutine could possibly be generalized. These retrieval problems are of the type in which a small amount of information is required from some SLR. Normally this information must meet comparison requirements to be printed. There are two forms: alphabetic and numeric retrieval. Alphabetic is the simpler of the two and has been described in example 3. The retrieval of a few fields of numeric data using RPG and a parameter card for a generalized subroutine is described in the following.

An assembler routine is presented in this appendix, listing L3, which will supply to an RPG program up to three fields from any SLR for

comparison and/or printing purposes. The file is defined in RPG, any calculation required is developed in RPG and all printing is formatted in RPG. In addition, the programmer must supply a parameter card which specifies the location and lengths of fields to be retrieved. In RPG an EXIT to the subroutine is written along with the names of fields to be supplied to and retrieved from this subroutine. With this the programming job is finished. He has written a complete program for a generalized routine.

The subroutine written is generalized. It modifies itself depending upon the values of the parameters supplied to it. It uses the parameters to recognize the SLR to be retrieved and the field locations for slotting for subsequent RPG use (for comparison and printing). The programmer need only supply the parameters and the routine will locate and slot the fields.

The assembler routine is shown in L3 of this appendix. It was used to produce the report L4 from the disk file L1. The RPG program written to retrieve the data from the file, and print the report upon certain comparison requirements is shown in L2.

The disk file L1 is similar to Example I, apendix I, L1, but has two fields per 02 SLR, one 7 digits and the other 4 digits long. The report requirements are: Retrieve the fields in positions 15-18 and 20-26 of SLR 02 for every record. Check the larger field against 11654.23. When one is found less than that value, print both fields from that SLR, add the smaller field to an accumulator and go on to the next record. Summarize the smaller field at the end of the report.

The RPG program does all the comparison and printing requirements. The parameter card is defined in the RPG input specifications as follows:

CCL	NAME	IDENTIFICATION
1- 2	SLRID	SLR identification code
3- 6	ALOC	location of field A
7- 8	ALEN	length of field A
9-12	BLOC	location of field B
13-14	BLEN	length of field B
15-18	CLOC	location of field C
19-20	CLEN	length of field C

In this example the parameter card had the following form:

02002007001504

02 - SLR 02

0200 - A field location (relative to the beginning of the SLR)

07 - A field length (number of digits to be printed)

0015 - B field location

04 - B field length

These parameters when supplied to routine GT SLR5 (RLABL SLRID) allowed the routine to retrieve the fields required. The fields had to be named A1-A15, B1-B15, C1-C15. This allows for the complete range of numeric fields handled by RPG 1 to 15 numeric digits and allows up to three fields per SLR. If any more are required from the same SLR, a complete SLR retrieval program should handle this request.

L3 is similar in many ways to the previous four examples. It

isn't significantly longer than the previous four L3's except in the data definition area where all the possible field lengths must be present. A program of this type involving similar parameter card approach should be produced to solve small numeric requests. Together with a routine for small alphabetic requests and one routine for each complete SLR retrieval most data retrieval problems can be solved.

```
** L 1 **          GDR  EXAMPLE 5  TEST  DATA

00001M01     02008003     04    J B MITCHEL          39 RIVER ROAD          1
00001020160    1245 4795246                                                 2
00001020240    3265 6543210                                                 3
00001020320    5689 2000101                                                 4
0000102        2441 0034765                                                 5
00002M01     02008003     04    R TOMPSON            656 NORTH DRIVE        1
00002020160    2255 4795246                                                 2
00002020240    5555 6543210                                                 3
00002020320    6682 2000101                                                 4
0000202        5439 0039287                                                 5
00003M01     02008003     04    L SMALLMOUTH         17 ELDER AVENUE        1
00003020160    6875 4795246                                                 2
00003020240    3256 6543210                                                 3
00003020320    2456 2000101                                                 4
0000302        6897 0046583                                                 5
00004M01     02008003     04    ZEKE HANDLE          99  BUXTON ROAD        1
00004020160    1245 4795246                                                 2
00004020240    3568 6543210                                                 3
00004020320    2211 2000101                                                 4
0000402        5885 0092162                                                 5
00005M01     02008003     04    ARNOLD GLERK         69 RUE MARIE           1
00005020160    9111 4795246                                                 2
00005020240    5513 6543210                                                 3
00005020320    3564 2000101                                                 4
0000502        2578 0069387                                                 5
```

** L 2 ** GDR EXAMPLE 5 RPG PROGRAM

```
00000H                                                                  EX5
01010FPARMCARDIP  F  80  80           READ40 SYSIPT                     EX5
01020FMEDRE   ISEAF 400 400           DISK11 SYS007S                    EX5
01030FAREPORT O   F 132 132     OF    PRINTERSYSLST                     EX5
02010IPARMCARDAA  01  80 CP                                             EX5
02020I                                      1   2 SLRID                 EX5
02030I                                      3   6 ALOC                  EX5
02040I                                      7   8 ALEN                  EX5
02050I                                      9  12 BLOC                  EX5
02060I                                     13  14 BLEN                  EX5
02070I                                     15  18 CLOC                  EX5
02080I                                     19  20 CLEN                  EX5
03010IMEDRE   BB 05   6 CM                                              EX5
03020I                                      1   5 KEY                   EX5
03030I                                      7  30 XYTAB                 EX5
03040I                                     31  48 NAME                  EX5
03050I                                     49  64 ADDR                  EX5
03060I                                      1 200 REC                   EX5
03070I                                    201 400 REC2                  EX5
04009C   01               GOTO BYPASS                                   EX5
04010C            LOOP    TAG                                           EX5
04020C                    SETOF                    020313               EX5
04030C                    EXIT GTSLR5                                   EX5
04040C                    RLABL          REC                            EX5
04050C                    RLABL          IN02                           EX5
04060C                    RLABL          IN03                           EX5
04070C                    RLABL          IN13                           EX5
04071C                    RLABL          SLRID                          EX5
04080C                    ULABL          A7      72                     EX5
04090C                    ULABL          B4      41                     EX5
04100C   02       A7      COMP 11654.23            070607               EX5
04110C   02 07             GOTO LOOP                                     EX5
04120C   02 06    SUM      ADD  B4        SUM     102                   EX5
04130C            BYPASS   TAG                                          EX5
04140C                     SETOF                    01                  EX5
050100AREPORT H  201   OF                                               EX5
050200        OR         1P                                             EX5
050300                                 50 @GDR TEST PROGRAM NO. 5 @     EX5
050400                                 75 @SEARCH FOR ONE A AND B @     EX5
050500                                 95 @PER LOGICAL RECORD@          EX5
050600         H  1      OF                                             EX5
050700        OR         1P                                             EX5
050800                                 10 @KEY@                         EX5
050900                                 40 @NAME@                        EX5
051000                                 70 @ADDRESS@                     EX5
051100                                100 @FIELD A@                     EX5
051200                                110 @FIELD B@                     EX5
051300         D  1      05                                             EX5
051400                          KEY    10                               EX5
051500                          NAME   50                               EX5
051600                          ADDR   75                               EX5
051700                    02    A7    100 @     0.  @                   EX5
051800                    02    B4    110 @  0. @                       EX5
060100         D  1      05 03                                          EX5
060200                                 30 @VALIDITY ERROR@              EX5
060300         D  1      01 13                                          EX5
060400                                 30 @INVALID PARMATER CARD ID@    EX5
```

** L 2 ** GDR EXAMPLE 5 RPG PROGRAM

```
060500        T  2     LR                                                  EX5
060600                                    30 @SUMMARY OF FIELD B4@         EX5
060700                          SUM       50 @      0. @                   EX5
```

```
                ** L 3 **          GDR  EXAMPLE 5  ASSEMBLER  SUBROUTINE

RPGSUB5  START 0
         ENTRY GTSLR5,A1,A2,A3,A4,A5,A6,A7,A8,A9,A10,A11,A12,A13,A14
         ENTRY A15,B1,B2,B3,B4,B5,B6,B7,B8,B9,B10,B11,B12,B13,B14,B15
         ENTRY     C1,C2,C3,C4,C5,C6,C7,C8,C9,C10,C11,C12,C13,C14,C15
         EXTRN REC,IN02,IN03,SLRID,IN13
GTSLR5   BALR  15,0
         USING *,15
         B     BEGIN
SAVREG   DS    16F
ZERO     DC    C@    @
A1       DC    PL1@0@     FIELDS OR SLOTS THAT WILL CONTAIN VALUES FROM
A2       DC    PL2@0@               SLR AREA TO BE  RETURNED
A3       DC    PL2@0@                  TO  RPG  BY  NAME
         DS    BL1
A4       DC    PL3@0@
         DS    BL1
A5       DC    PL3@0@
         DS    BL2
A6       DC    PL4@0@
         DS    BL2
A7       DC    PL4@0@
         DS    BL3
A8       DC    PL5@0@
         DS    BL3
A9       DC    PL5@0@
         DS    BL4
A10      DC    PL6@0@
         DS    BL4
A11      DC    PL6@0@
         DS    BL5
A12      DC    PL7@0@
         DS    BL5
A13      DC    PL7@0@
         DS    BL6
A14      DC    PL8@0@
         DS    BL6
A15      DC    PL8@0@
B1       DC    PL1@0@
B2       DC    PL2@0@
B3       DC    PL2@0@
         DS    BL1
B4       DC    PL3@0@
         DS    BL1
B5       DC    PL3@0@
         DS    BL2
B6       DC    PL4@0@
         DS    BL2
B7       DC    PL4@0@
         DS    BL3
B8       DC    PL5@0@
         DS    BL3
B9       DC    PL5@0@
         DS    BL4
B10      DC    PL6@0@
         DS    BL4
B11      DC    PL6@0@
         DS    BL5
```

```
B12     DC    PL7@0@
        DS    BL5
B13     DC    PL7@0@
        DS    BL6
B14     DC    PL8@0@
        DS    BL6
B15     DC    PL8@0@
C1      DC    PL1@0@
C2      DC    PL2@0@
C3      DC    PL2@0@
        DS    BL1
C4      DC    PL3@0@
        DS    BL1
C5      DC    PL3@0@
        DS    BL2
C6      DC    PL4@0@
        DS    BL2
C7      DC    PL4@0@
        DS    BL3
C8      DC    PL5@0@
        DS    BL3
C9      DC    PL5@0@
        DS    BL4
C10     DC    PL6@0@
        DS    BL4
C11     DC    PL6@0@
        DS    BL5
C12     DC    PL7@0@
        DS    BL5
C13     DC    PL7@0@
        DS    BL6
C14     DC    PL8@0@
        DS    BL6
C15     DC    PL8@0@
PACKED  DS    D
SW      DC    F@0@            INTERNAL SWITCH FOR PARM FIELDS.
BEGIN   STM   0,15,SAVREG
        LM    2,5,#A%IN02,IN03,REC,SLRID¤
        USING RECORD,4
        L     13,#A%IN13¤
        LR    9,4                   REG9 HAS  REC ADDRESS NOW
LOCATE  A     9,#F@6@
        CLC   0%2,9¤,0%5¤           COMPARE XYTAB ID TO PARMATER CARD ID
        BL    LOCATE
        BH    ERR2                  PARAMETER CARD ID INVALID
        A     9,#F@2@               REG9 POINTS TO SLR CHAINING ADDRESS
        MVC   SW,#F@0@
MOVE    LR    8,4                   REG8  POINTS TO REC
        PACK  PACKED,0%4,9¤         PACK SLR CHAINING ADDRESS
        CVB   6,PACKED
        AR    8,6                   REG8 POINTS TO SLR REQUIRED
        CLC   0%5,8¤,KEY
        BNE   ERR
        CLC   5%2,8¤,0%5¤
        BNE   ERR
SETUP1  LA    11,A1                 SET UP FIELD LENGTHS FIRST
SETUP2  PACK  PACKED,2%4,5¤
```

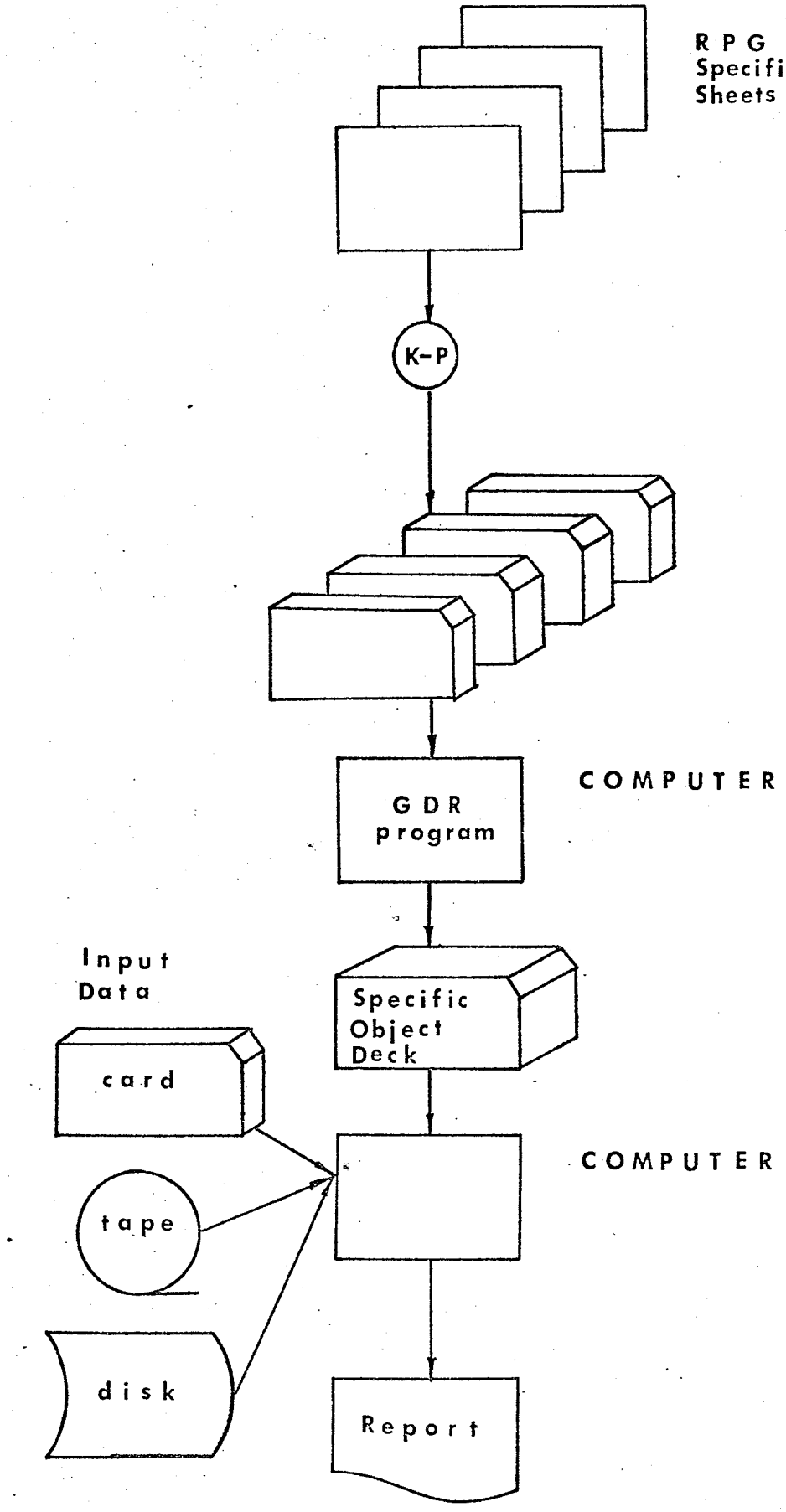
** L 3 ** GDR EXAMPLE 5 ASSEMBLER SUBROUTINE

```
         CVB   6,PACKED
         S     6,#F@1@          REG6 HAS DYSPLACEMENT FROM START OF SLR
         AR    6,8                   REG6 POINTS TO PARM SPECIFIED FIELD
         PACK  PACKED,6%2,5¤
         CVB   7,PACKED               REG7 HAS FIELD LENGTH IN ZONED DEC
         SRA   7,1                    REG7 HAS FIELD LENGTH IN PACKED DEC.
         LR    10,7
         SLA   10,4
         CVB   7,PACKED
         S     7,#F@1@
         OR    10,7                        REG10 HAS L1 AND L2 FOR REMOVE INST.
SETUP3   L     12,#F@1@           LOCATE  CORRECT PACKED FIELD FOR
         CVB   7,PACKED                      INSERTION AND RETURN
LOOP     CR    12,7                             TO RPG
         BE    EXECUT
         AR    11,12              UPDATE SLOT ADDRESS BY COUNT
         A     12,#F@1@
         B     LOOP
EXECUT   EX    10,REMOVE          PACK SLR FIELD INTO SLOT
         A     5,#F@6@
         CLC   2%4,5¤,ZERO
         BE    NEXTSLR            NO MORE FIELDS TO BE RETRIEVED BY PARM.
         CLC   SW,#F@0@                        CARD SPECS.
         BE    SETB1              B FIELD TO BE RETRIEVED
         CLC   SW,#F@1@
         BE    SETC1              C FIELD TO BE RETRIEVED
REMOVE   PACK  0%0,11¤,0%0,6¤          MOVE SLR FIELD INTO SLOT
SETB1    LA    11,B1
         MVC   SW,#F@1@
         B     SETUP2
SETC1    LA    11,C1
         B     SETUP2
NEXTSLR  MVC   0%4,9¤,7%8¤        LOAD CHAIN ADDRESS FROM SLR INTO XYTAB
         MVI   0%2¤,X@F0@         SET ON IN02 SINCE A FIELD IS LOCATED
         B     RETURN
ERR      MVI   0%3¤,X@F0@         VALIDITY ERROR
         B     RETURN
ERR2     MVI   0%13¤,X@F0@        INVALID PARM CARD ID
RETURN   LM    0,15,SAVREG
         BR    14                 RETURN  TO RPG
RECORD   DSECT
KEY      DS    CL5
         END
```

GDR TEST PROGRAM NO. 5 SEARCH FOR ONE A AND B PER LOGICAL RECORD

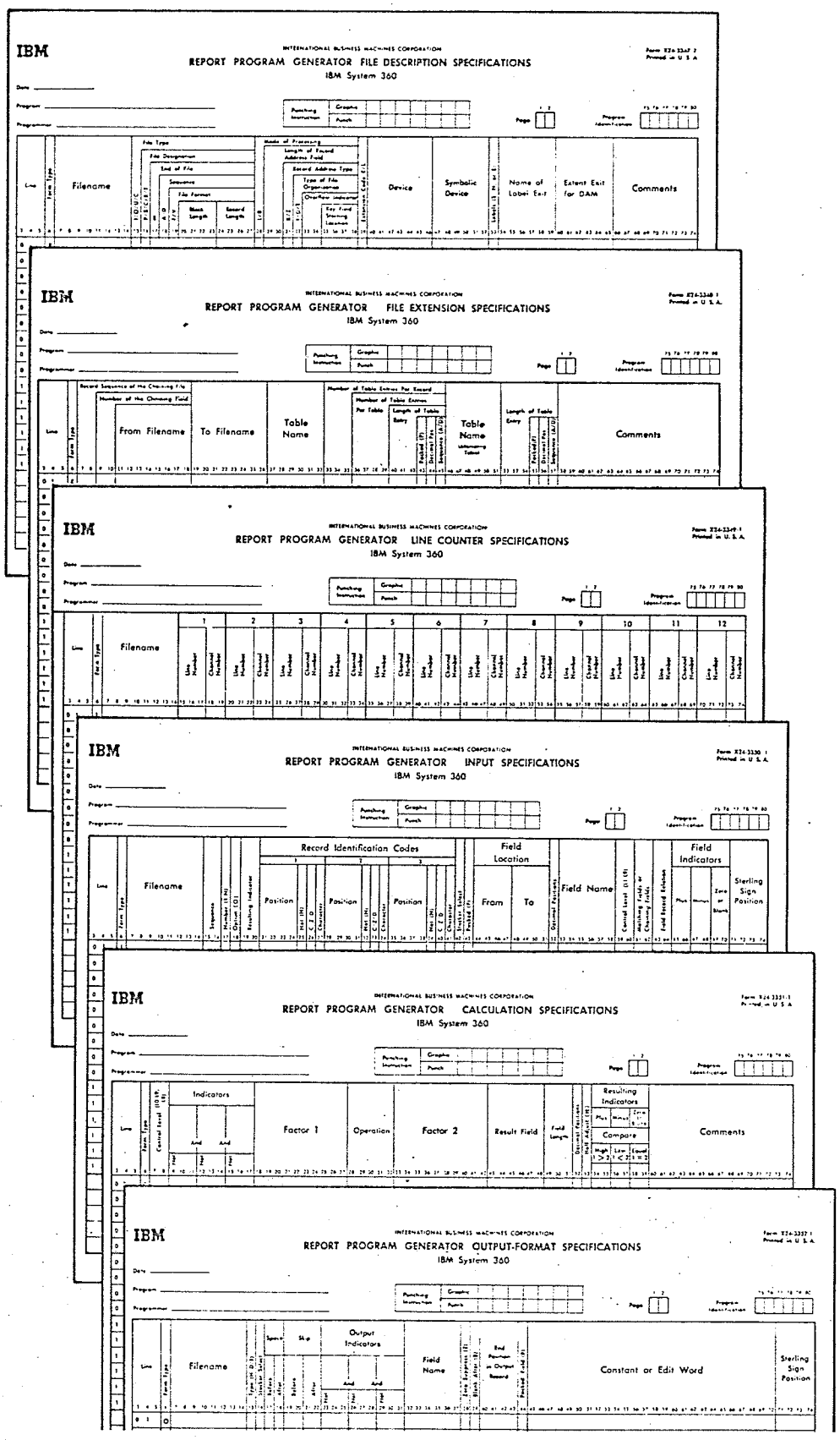
KEY	NAME	ADDRESS	FIELD A	FIELD B
00001	J B MITCHEL	39 RIVER ROAD	347.65	244.1
00002	R TOMPSON	056 NORTH DRIVE	392.87	543.9
00003	L SMALLMOUTH	17 ELDER AVENUE	465.83	689.7
00004	ZEKE HANDLE	99 BUXTON ROAD	921.62	588.5
00005	ARNOLD GLERK	69 RUE MARIE	693.87	257.3
SUMMARY OF FIELD B4	2324.00			

Figure 1

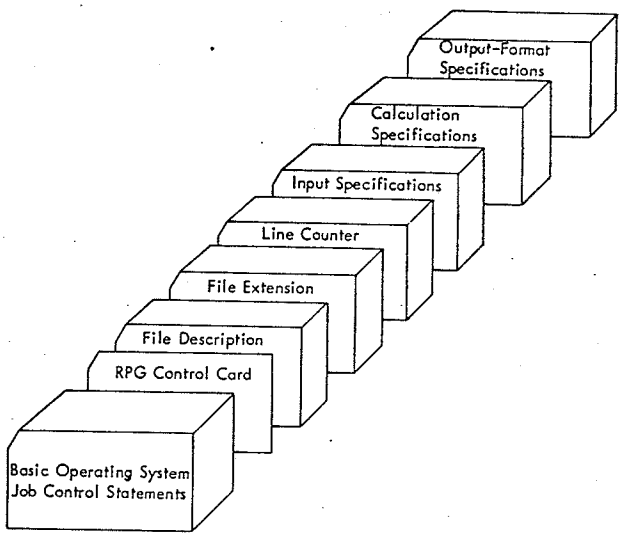


Producing Reports Using The RPG Program

Figure II

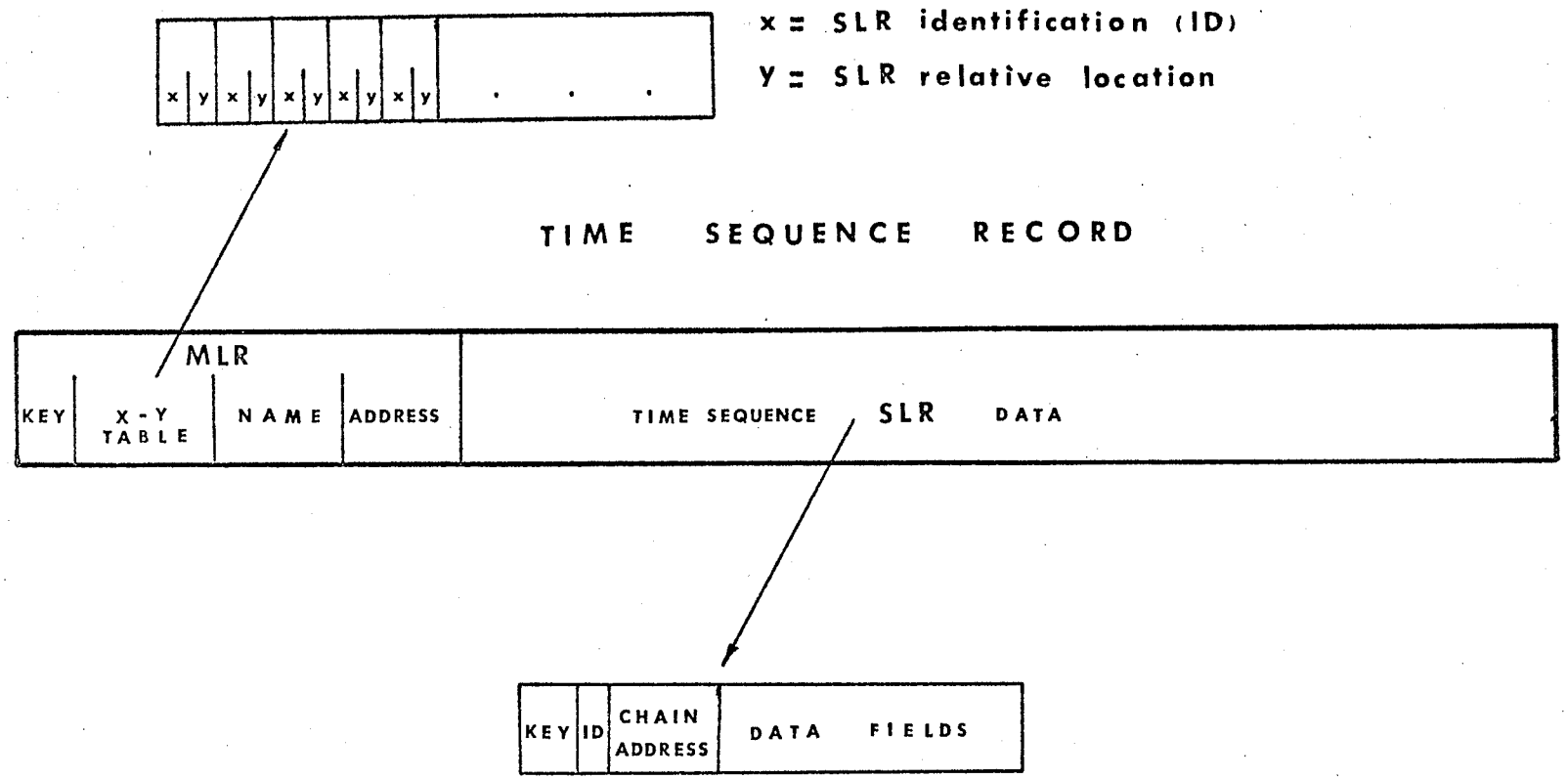


RPG Specifications Sheets



RPG Deck Arrangement in the Basic Operating System Input Stream

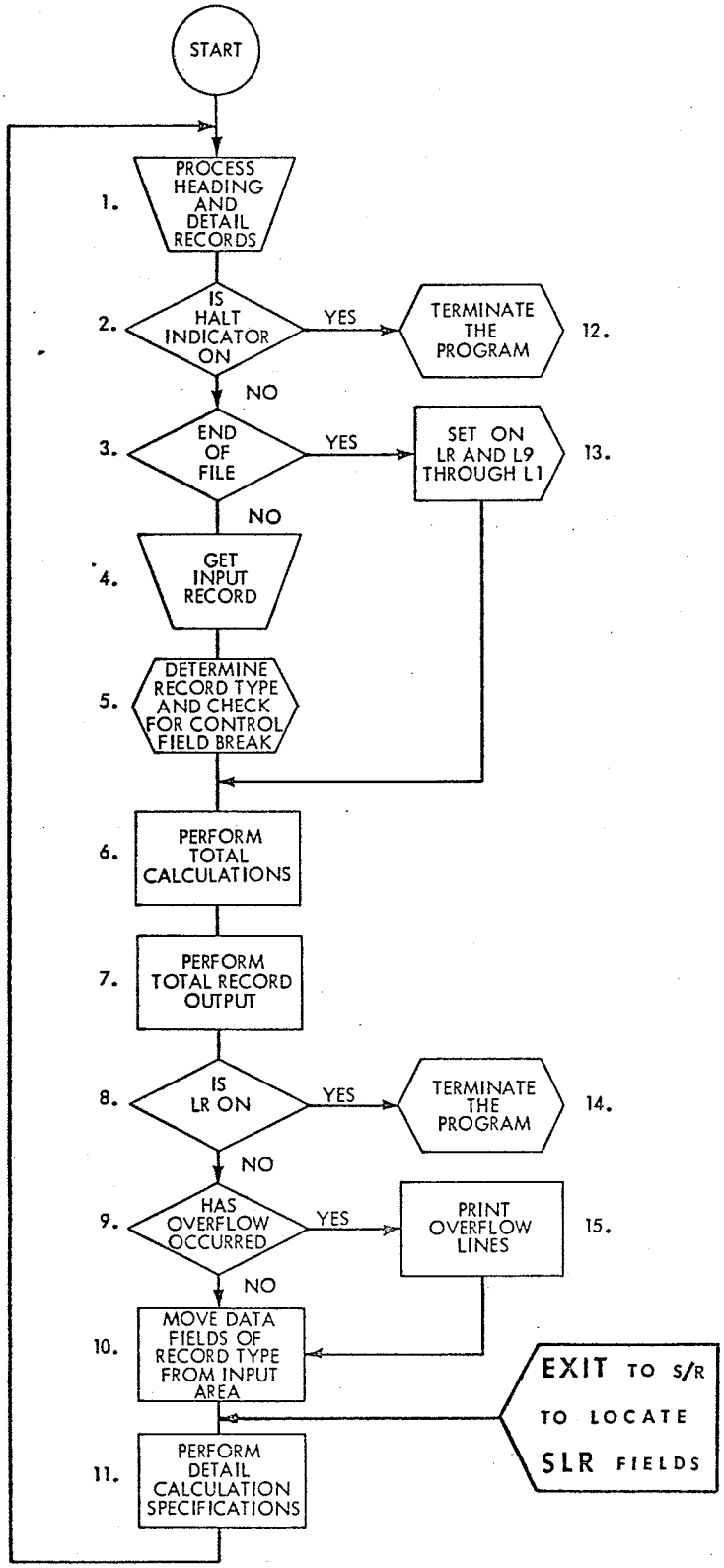
Figure IV



General Format of the Time Sequence Record

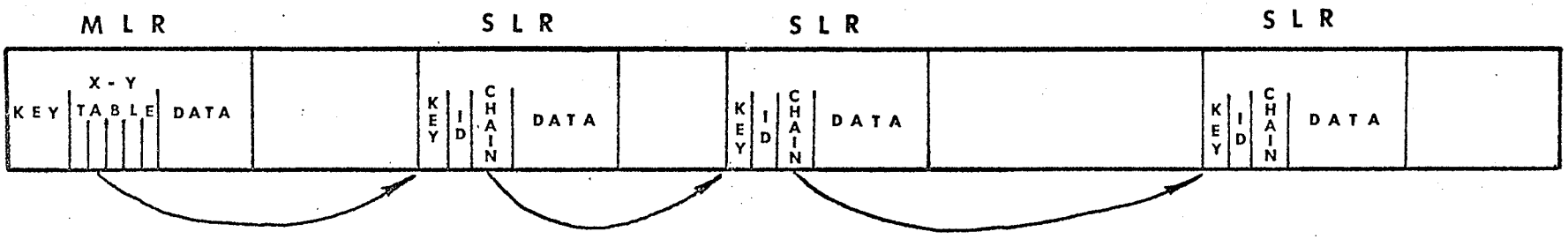
Figure V

PROGRAM LOGIC



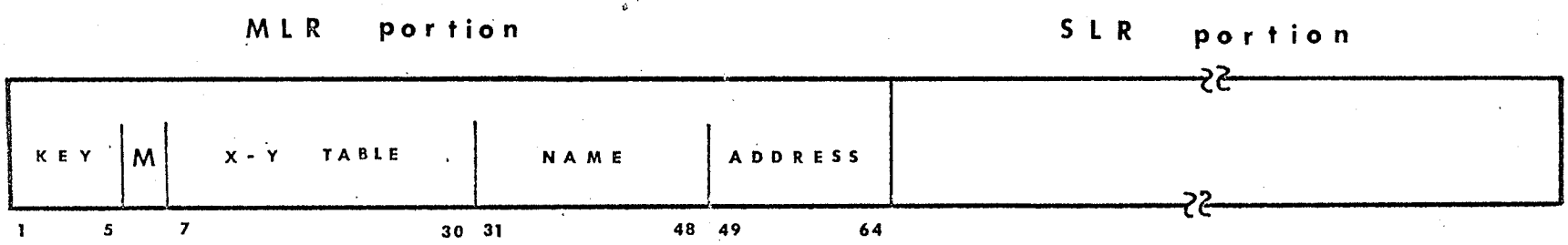
General Logic Flow of an RPG Object Program

Figure VI



Time Sequence Record Chaining Technique

Figure VII



Time Sequence Record Format For

Examples I,II,III,IV

Figure VIII

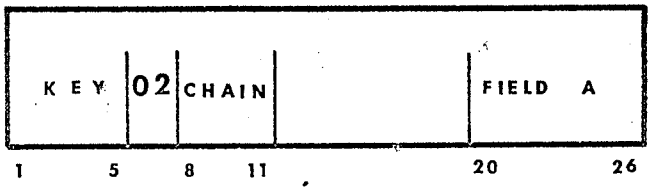


Figure IX

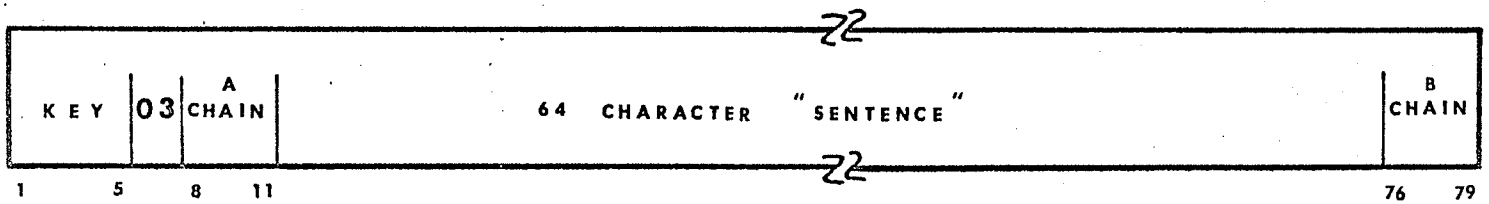
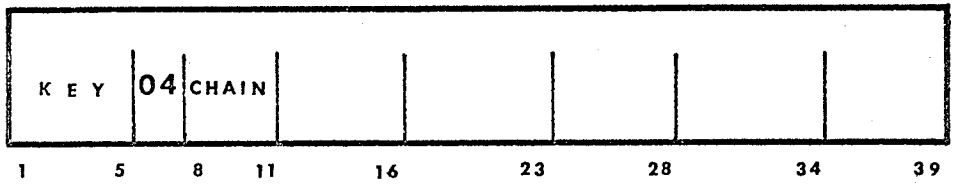


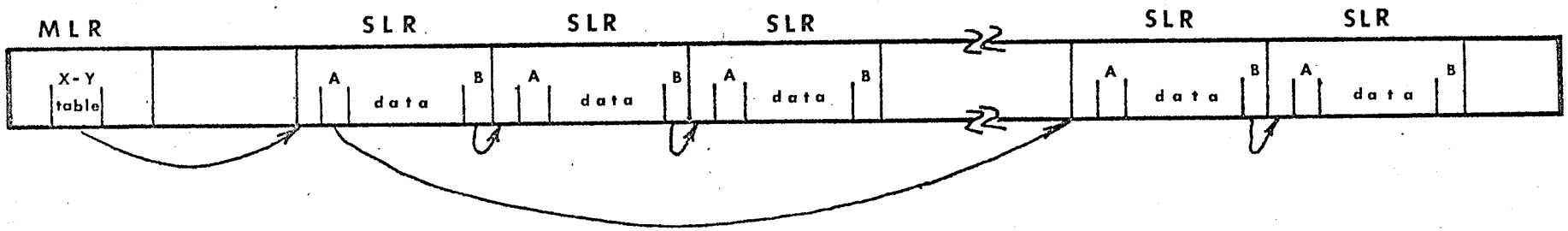
Figure X



SLR Format For Examples I,II,III,IV

Figure XI

TIME SEQUENCE RECORD



Example III Sentence and Paragraph Chaining Technique

GLOSSARY OF TERMS

Assembler routine	- program written in symbolic language which is closest to machine language.
Auxiliary storage	- computer disk, drum, or tape.
BALR	- /360 assembler language macro.
BOS	- Basic Operating System /360.
Chain field	- data used in a record to locate similar records in auxiliary storage.
Cobol	- Common Business Oriented Language.
CPU	- Central Processing Unit.
DP	- Data Processing.
DOS	- Disk Operating System.
EDP	- Electronic Data Processing.
ENTRY	- /360 assembler language macro.
EXIT	- /360 RPG language macro.
EXTRN	- /360 assembler language macro.
Field	- smallest element of information.
File	- logical grouping of records.
GDR	- generalized data retrieval - programming technique in which one data retrieval program is written and modified by parameter cards to suit a specific application.
ID	- identification.
IOCS	- Input Output Control System.
KEY	- code identifying a logical record.
Language translator	- manufacturer written program which converts symbolic language into machine language.

Linkage	- programming routine used to connect two programs together.
MACRO	- a programming word which is expanded into machine language by the corresponding compiler.
MLR	- Main Logical Record.
Parameter card	- data on cards which modify program logic.
Real Time	- a concept used in computing in which the computer immediately affects the environment which creates data for it.
Record	- logical grouping of fields.
RLABL	- /360 RPG language macro.
RPG	- Report program Generator - a programming system developed by IBM for their /360.
Slot	- storage location of a field of data.
SLR	- Secondary Logical Record.
Spec.	- specification.
START	- /360 assembler language macro.
TAG	- /360 RPG language macro.
Tailor-made program	- a program written for a particular application.
Time Sequence File	- a type of data organization in auxiliary storage in which the data is present in the order in which it was received. Today's data before tomorrow's.
TOS	- Tape Operating System /360.
ULABL	- /360 RPG language macro.
USING	- /360 assembler language macro
Validity check	- program logic to ensure correct data is being received.

BIBLIOGRAPHY

1. Bachman C. W., "Software for Random Access Processing" Datamation, April 1965, pages 36-41.

2. Canning, Richard G., "File Management and Data Retrieval" EDP Analyzer, Novermber, 1964, Vol 2, No. 11.

3. _____ "How to Organize Files", EDP Analyzer, October, 1964, Vol 2, No. 10.

4. _____ "New Approaches to Random Access Files" EDP Analyzer, May, 1965, Vol 3, No. 5.

5. Davies, G., "General Information Retrieval and Listing System" Imperial Oil Limited, Calgary, Alberta, K.R. Marble, Manager, Systems and Computer Services, Western Region.

6. General Electric Computers, Compatables /400, Report Program Generator Manual, CPB-3588

7. Henderson, P. B., "On the Design of Data Systems for File Analysis and Information Retrieval" Paper presented to The Institute of Management Sciences, Eleventh Annual International Meeting, Pittsburgh, Penn., March 12-14, 1964.

8. _____ "A Computer-Generated Thesaurus" Paper presented to the Joint CORS-ORSA Conference - Montreal, Canada, May 27-29, 1964.

9. _____ "System Specifications for Data Retrieval" Operations Research Society of America, Western Section, First International Meeting, Honolulu, Hawaii, September 14-18, 1964.

10. _____ "Computer Concepts for Management - DATA RETRIEVAL" , A paper presented to Harvard University, July 14, 1965.

11. Honeywell Series 200, Report Generator Manual, DSI-325-A, 5465, Honeywell Inc., Electronic Data Processing Division, Wellesley Hills, Massachusetts 02181.

12. IBM System /360 Reference Manuals:

(1) Report Program Generator

(16 K DISK OR TAPE)

FILE: S360-28

FORM: C26-3570-2

(2) Principles of Operation

FILE: S360-01

FORM: A22-6821-1

(3) Assembler Language

FILE: S360-21

FORM: C28-6514-3

(4) DOS, System Control and System Service Programs

FILE: S360-36

FORM: C24-3428-1

13. IBM System /360 Newsletter, Subroutines of RPG programs, Toronto Test Centre, by L. J. Sabo.

14. Kosakoff M. and Buswell D.L., "Experience with a Generalized Information Processing System", Proceedings - Fall Joint Computer Conference, 1963, pages 183-192.

15. McGee, W.C., "Generalization: Key to Successful Electronic Data Processing" A.C.M. Journal, January, 1959, pages 1-23.

16. McGee, W.C. and Tellier H., "A Revaluation of Generalization," Datamation Vol 6, No. 4, 1960, pages 25-29.

17. Pan, G.S., "Generalized Structure and Optimum File Design Considerations", paper presented at the Fifth Annual Meeting of The Computer Society of Canada, May 30, 1966 (as yet unpublished).

18. Postley, J.A., "File Management: Generalized Application Programs", Publication of Informatics Inc., 15300 Ventura Boulevard, Serman Oaks, California 91403.

19. Postley, John A. and Buetell, T.D., "Generalized Information Retrieval and Listing System" Datamation, Vol 8, No. 12, 1962 pages 22-25.

20. The Financial Post

Introduction - February 5, 1966 page 1.

Part II, Chapter 4 - May 21, 1966 page E-26.

21. Tiwari, R.K. unpublished thesis presented to the Department of Computing Science, University of Manitoba 1967.

22. Watt, L.J., "Generalized Edit and Maintenance" , Imperial Oil Limited, Calgary, Alberta.

23. Woods, Gordon & Co., "1966 Guide to Data Processing Equipment", A survey by Woods Gordon and Co., Management Consultants, 15 Wellington Street, West, Toronto.