Principles of Programming

Section 1: The Nature of Data Processing

IBM Personal Study Program

The IBM Personal Study Program offers the opportunity to develop an understanding of and an appreciation for the tools of data processing, their operations and application. In recent years, the use of automatic data processing equipment has been extended into almost every area of business, government and science. As a result, the need for people knowledgeable in the subject has multiplied manifold—and is continuing to multiply.

The purpose of the IBM Personal Study Program is to help satisfy this need by providing simplified self-study texts covering the fundamentals of data processing. With the background these texts provide, the interested student will be prepared to delve further into those areas of greatest interest to him and his career.

Section 1: The Nature of Data Processing

1.1 Introduction

This book is intended to provide a basic understanding of what is required to make an electronic computer do useful work. In order to reach such an understanding, it is necessary to discuss such topics as:

1. What kinds of things can computers do in business data processing?
2. What is required to specify the processing to be done?
3. What are the components of a computer, what are their functions, and how do they work?
4. What is "programming," and what are the important programming techniques and principles?

Of these four areas, the emphasis in this book is largely on the last. The other questions are treated briefly in the first two sections and indirectly throughout the book. The first two questions relate more directly to the subject of *system analysis* or *procedure design*. They are crucial to the successful utilization of an electronic data processing system, but cannot be studied properly without the background of an understanding of programming. The third question relates to computer engineering; the programmer must know a few broad, general characteristics of the subject, but almost none of the detail.

It is assumed that the reader of this book has had no experience with computers or with punched cards. No knowledge of mathematics or accounting is required.

The reader who completes a careful study of this book may expect to know a good deal about programming. He will know all the basic principles of the subject, a little of how to get started on a project, and what the major steps are. He will find it much simpler to learn programming for another computer, or to proceed with a detailed study of the machine (the IBM 1401) which is sketched here. He should not expect, however, that he will know enough to undertake by himself the programming of a major application. Learning programming requires a certain amount of practice, and, ideally, an opportunity to work on one or two applications with an experienced person.

The programming ideas presented here are illustrated in terms of the IBM 1401 Data Processing System. This machine was chosen primarily because of its wide distribution. It should be realized, however, that most of the basic principles of programming are applicable to any

computer. The reader who learns the material in this book thoroughly will have relatively little difficulty learning any other machine. There are a few features of the IBM 1401 that are not strictly typical of all computers, but they constitute a small part of the subject when compared with the broad general principles. It should also be recognized that *every* computer has specialized features. No one should be concerned that he is learning material here which will not be useful to him. Finally, it should be noted in this connection that no attempt is made to cover all of the features of the IBM 1401 system; this book should be regarded as an introduction to programming, not as a manual of programming for the IBM 1401.

One last matter of information about the book itself: The review questions and exercises are important. The review questions, which appear at the end of most subsections, let the reader assure himself that he understands the material thoroughly before proceeding. If the material has been understood, these questions will be relatively easy; if they seem difficult, a rereading would probably be a good idea. The exercises at the end of each section provide an opportunity to apply what has been studied. In some cases they continue the development of a topic that could not be treated fully in the text for lack of space. Answers to some exercises are given in Section 12.

1.2 Basic Data Processing Ideas

Electronic computers are used in business for a variety of reasons. When properly applied, they can save time or money (or both) in producing reports for management and government, in preparing checks and earnings statements for employees, in writing statements to customers, in keeping records of accounts payable to suppliers. In many situations they make it possible to obtain information that would otherwise not be economically justifiable. In some cases they provide the basis for an improved management control of a business that would not be feasible for time or money reasons without a computer. They are also widely used for engineering and scientific computations.

In carrying out these functions, a number of basic computer operations are performed. Information appearing on punched cards is *listed* (printed). Various *calculations* are performed on data. Detailed information is *summarized* (totaled), often according to several classifications. Information is *edited*, which means two rather different things. In one meaning, *source data* (input information) is checked for validity and accuracy before it is used in further processing. In the other meaning, editing refers to the rearrangement of results for easy reading, such as by inserting dollar signs, decimal points and commas, deleting zeros in front of numbers, and providing adequate space between numbers.

These *operations* are performed on *data*. It is necessary also to consider how the data is organized, since the arrangement of the information has a most significant effect on how the processing is done. This brings us to one of the most fundamental concepts in data processing, that of a *file*.

A file is a collection of *records* containing information about a group of related accounts, people, stock items, etc. For instance, an accounts receivable file contains a record for each customer, showing at least the customer's name, address, account number, and amount owed. It may also contain his credit limit, the length of time the amount owed has been due (the "age" of the account), and other information, depending on the needs of the particular business. In a payroll file, the record for each employee contains such information as name, payroll number, department, sex, Social Security number, number of dependents, pay rate, year-to-date gross earnings, year-to-date taxes withheld, year-to-date Social Security tax, and often many other things.

These examples both involve *master files*, which contain semipermanent information, some of which is *updated* (modified) periodically. A *transaction file*, on the other hand, contains information used to update a master file. Examples: a file containing a record for each customer purchase, or a file of labor vouchers used to calculate gross pay. In addition to master and transaction files there are *report files*, which contain information extracted from a master file. An example is provided by the quarterly Social Security reports required by the federal government.

It is obviously necessary to have some way to identify each record in a file. This is usually accomplished by establishing one item in the record as the *key* or *control field* of the record. The key distinguishes each record from all others, and is used in almost all file operations. Examples of keys: the customer's account number in an accounts receivable application, the employee's pay number in a payroll, the part number in an inventory control application, the salesman's number in a sales commission job.

Almost all data processing involves operations on files. It is frequently necessary to *sort* the records in a file, that is, to put the records into ascending sequence (or descending, sometimes) according to the keys of the records. For example, it may be necessary to sort employee labor vouchers into sequence on payroll number before this transaction file can be processed against the payroll master file. As we shall see later in this section, data processing methods fall into two broad and rather different classes, according to whether the files involved are or are not required to be sorted before the primary processing can be done.

Another common file operation is the combining of two or more files to form one file. If the combined file contains all records from the sep-

arate files, this operation is called *merging*; if some of the original records are omitted from the combined file it is properly called *collating*. (The distinction between the two terms is not always observed, in practice.)

Careful planning is required to combine the basic operations so that the files are properly processed and the desired results produced. It is necessary to establish goals of the application, the time schedules that must be met, the exact nature of the operations to be carried out, etc. All of this takes more time than might first be thought, for two reasons that are fundamental to a proper understanding of electronic data processing.

1. All processing must be defined in advance, with a very few exceptions. For instance, it often happens that a customer sends in a check for an amount different from the amount shown on his bill. The person planning the accounts receivable job cannot proceed on the assumption that all payments will be for exact billed amounts and say, "I'll worry about that problem when it happens." The processing operations for such a situation must be planned *in advance*. Again, it is necessary to decide what to do about possible processing errors *before* an application is placed on the machine.

2. A machine cannot exercise judgment unless it has been given explicit directions for making a decision. A machine can be set up to make relatively complex decisions, if they are expressible in quantitative terms, but it must be *instructed* how to make the decisions and what to do in each alternative. We can say to a computer, in suitable language, "If a man's deductions exceed his gross pay, omit as many deductions as necessary; the order in which to omit them is specified in the following table, where the first deduction is the least crucial." We *cannot* say, "If anything unusual comes up, do what you think is best."

When the task has been properly defined in terms of *what* is to be done, the next step is to decide *how* to do it with the computer. This step involves expressing the processing in terms of operations that can be carried out with the available computing equipment. One of the primary tools of this step is the *flow chart*, which shows the sequence of operations in a graphic form. Several flow charts appear in Sections 1.3 and 1.4.

The next step is *programming*. This includes two activities, one of which is *block diagramming*. A block diagram is a detailed flow chart, showing in greater depth exactly what is to be done at each stage of the computer processing. The other activity in programming is *coding*, which is the primary subject of this book.

The fundamental problem is this: The "language" in which the computer can accept "instructions" is very different from the language in which we ordinarily describe data processing. One way or another, the procedure to be followed must be translated into the computer's language. For instance, *we* say, "Summarize sales by salesman and district." The computer understands only instructions like "Add these two numbers," or "Go to the print steps if these two numbers are not the same," or "Read a card and place the information in the card input area."

Coding is the process of stating a procedure in a language acceptable to the computer. (The word comes from the fact that the computer's basic language consists of instructions that are written in a "coded" system of numbers and letters.) In few cases are we required to do the entire job of translation, all the way to the final form of the instructions as they will be obeyed ("executed") by the computer. Usually, we write instructions in a symbolic form that is rather similar to the machine's language, but considerably more convenient for us. The final step of the translation is then performed with the machine's assistance. In other situations we are able to write the machine procedure in a language quite similar to ordinary English, with the bulk of the translation being done with the aid of a special computer program (set of instructions).

Programming and coding involve so much detailed work that most programs do not operate correctly when first tried. Thus it is necessary to *debug* the program (locate and correct the errors) and to *test* it with *test cases* to be sure that it properly processes the data. All of this goes under the name of *program checkout*.

One more activity remains before the program is ready to be used: the master file must be prepared. This usually requires converting the file from the form in which it was used with the previous manual methods. File conversion can be a sizable task in itself, and one that often must be started well before the program is completed.

Review Questions

1. *What are some of the basic data processing operations?*
2. *Give an example of source data.*
3. *What is a file? Record? Master file? Transaction file? Give examples.*
4. *Define sorting and merging.*
5. *A computer can make decisions, under certain circumstances. Give two qualifications to this general statement.*
6. *What is the difference between programming and coding? Between flow charting and block diagramming?*

1.3 An Example of Sequential File Processing

Some of the ideas introduced above may be clarified by considering a typical example of data processing.

A certain company has a system of sales districts, each district having several salesmen. For each sale a transaction record is prepared, showing:

1. Product number
2. Quantity
3. Salesman number
4. District number

These transaction records are prepared in the form of punched cards at the data processing center, from reports sent in from the districts. Records such as these, with certain other information included, would ordinarily start a whole chain of data processing: instructions would be prepared for the shipping department; the customer's account would be charged with the amount of the purchase, less discounts; the inventory file of finished goods would be updated. For our purposes here, however, we shall consider only one aspect of the total data processing activity based on these records: the preparation of sales statistics.

For various purposes it is desirable to obtain sales figures summarized monthly in several classifications:

1. Total sales of each merchandise item for the month.
2. Total sales of each salesman for the month.
3. Total sales of each district for the month.
4. Total sales for the company for the month.

Before outlining the sequence of operations required to produce these reports, we must consider how the characteristics of the master file affect the planning. The master file for this simplified problem is needed only to get the unit price for each product; in a full-scale application it would do much more. Our master file contains a record for each product, showing product number and the price of one unit. The most important consideration for our purposes is that the file is in product number order. This means that the record for the product with the smallest product number is first in the file, the record for the product with the next larger product number is next, etc.

The first computation will be to get from the master file the unit price of each product sold. This means that for each transaction record we must look up the unit price, which could be done by searching the master file once for each transaction record in order to find the price for that item. This can be done if the master file is stored in a form that makes it convenient to get at any record with a minimum of delay; this is the subject of the next section. However, when the master file is stored on cards or magnetic tape, this approach has serious drawbacks. The difficulty is that to find any one record in a file of cards (or on magnetic tape), it is necessary to inspect every record until the desired one is found. With the equipment involved, it is not possible to thumb through a deck of cards until the approximate area is located and then search in detail for the correct card; it is necessary to read every card in sequence.

This is the basis of the term *sequential access* file: each record is available only in the order in which it appears as the file is read *in sequence*. To be explicit, this means that the first record in the file is available with little delay, but to get the last record requires reading the entire file. This is contrasted with a *random access* file (see Section 1.4), where any record is available just about as quickly as any other.

The fact that our master file is of the sequential access type determines to a considerable extent how the processing must be done. We clearly do not want to have to read the master file once for each transaction card. Instead, we will put the transaction file into the same sequence as the master file, and then get all the information we need from the master file in one *pass*, that is, in one reading of it.

The resequencing of the transaction file is called *sorting*. Since our transaction file is a *deck* of cards, it can be sorted with a card sorter, as described a little more fully in Section 2.3. When this operation is completed, the deck of sales cards will be in the same sequence as the master deck. That is, the sales card with the smallest product number will be at the front of the deck, etc. There may be more than one card with the same product number, of course: several customers may have bought the same item.

Recall that what we are trying to do at this point is to get the unit price of each product from the master file, so that the total price of each sale can be computed (the sales cards give only the number of units sold). There are several ways of doing this; the choice depends on the characteristics of the computer to be used. We shall assume that this job is to be done on a computer that can read only one deck of cards and does not have magnetic tapes, but can separate the cards into two stacks after reading them. This corresponds to the equipment available on an IBM 1401 card system.

With the master and detail (transaction) decks now in the same sequence, we use the card *collator* to merge them. By this operation, each detail card is placed behind the master having the same product number. Several things can happen when this is done.

1. There may be *unmatched masters*—that is, masters for which there is no corresponding detail. This means simply that the particular product was not sold in the month. With the collator we have the choice of including such unmatched masters in the merged deck or of selecting them to fall into a separate pocket. We shall leave them in, to avoid having to put them back later. This will require a little extra work in the computer program, but nothing very difficult.

2. There may be *unmatched details* — that is, sales cards with product numbers not in the master file. This means that a product number was written or punched incorrectly. We cannot process such cards, so the collator must be set up to select them into a separate pocket. The error cards must be corrected and either reinserted in the deck or saved until the report for next month is run. We shall do the former. If there are only a few error cards, they can be inserted by hand; if there are many of them, another collator run can be used to insert them.

3. It could happen, through a mixup in card handling, that one or both of the decks are out of sequence. The collator can be set up to check for this possibility and stop if an error is detected. Such errors are not an everyday occurrence, but since they *can* happen and since they are fairly easy to check for, we may as well do so.

Before describing the rest of the procedure, we may review what has been covered so far, in terms of some sample data. In a typical application of this type, the master file would contain perhaps 10,000 records and there might be 20,000 sales cards in a month. The sample data is based on a master file of 16 records and also 16 sales cards, as shown in Figures 1a and 1b.

Master File

Product Number	Unit Price
1120	6.90
1190	4.32
1200	10.60
1213	25.50
1655	.80
1656	18.00
2441	2.57
2702	74.00
3495	2.70
4192	8.09
4377	21.90
4992	10.20
5009	8.00
5062	1.47
5100	7.75
5211	43.50

Figure 1a. Sample master file for sequential file processing example.

Detail File

Product Number	Quantity	Salesman	District
4992	8	31	3
4192	12	20	1
1190	55	32	3
1213	2	20	1
1655	80	31	3
4190	100	41	2
4992	11	10	1
5062	20	6	2
1655	20	6	2
1213	1	41	2
5062	75	32	3
1190	30	10	1
1190	16	61	3
1655	150	61	3
4192	7	32	3
1656	4	6	2

Figure 1b. Sample data (unsorted) for sequential file processing example.

The first step, sorting the detail deck, puts the details into the order shown in Figure 2. The merged deck is shown in Figure 3, where asterisks are written after the master file product numbers for clarity. Note that the master card appears in front of its associated detail cards.

Product Number	Quantity	Salesman	District
1190	55	32	3
1190	30	10	1
1190	16	61	3
1213	2	20	1
1213	1	41	2
1655	80	31	3
1655	20	6	2
1655	150	61	3
1656	4	6	2
4190	100	41	2
4192	12	20	1
4192	7	32	3
4992	8	31	3
4992	11	10	1
5062	20	6	2
5062	75	32	3

Figure 2. Sorted detail file for sequential file processing example.

When the two decks are merged, the sales card for product number 4190 will fall out as an unmatched detail. Suppose that investigation shows the product number to have been incorrectly punched; it should have been 1190. After the card has been repunched it can be placed anywhere in the group of sales cards for product 1190; it is shown in Figure 3 at the front of the group.

Product Number	Unit Price or Units Sold	Salesman	District
1120*	6.90		
1190*	4.32		
1190	100	41	2
1190	55	32	3
1190	30	10	1
1190	16	61	3
1200*	10.60		
1213*	25.50		
1213	2	20	1
1213	1	41	2
1655*	.80		
1655	80	31	3
1655	20	6	2
1655	150	61	3
1656*	18.00		
1656	4	6	2
2441*	2.57		
2702*	74.00		
3495*	2.70		
4192*	8.09		
4192	12	20	1
4192	7	32	3
4377*	21.90		
4992*	10.20		
4992	8	31	3
4992	11	10	1
5009*	8.00		
5062*	1.47		
5062	20	6	2
5062	75	32	3
5100*	7.75		
5211*	43.50		

Figure 3. Merged deck for sequential file processing example.

Now we are ready for the first calculation step. We must *get* the unit price of each product, *extend* the price for each sale (multiply unit price by the number of units sold), summarize the total sales dollars for each product (add up the price of each sale), and summarize all sales for the month. This is now easily done. As the combined deck is read by the computer, the unit price for each product can be obtained from the master card and stored for computing the price of each sale. As each sales card is read, the number of units can be multiplied by the unit price and this sale price added to the total of sales for the product. For later operations, a new detail card must be punched which contains all of the old information plus the extended price of each sale. When all of the sales cards for one product have been read and extended, the total sales for that product must be printed. When all cards have been read, the total sales for the month should be printed. Finally, as the cards are read, the original details (which may now be discarded) should be stacked separately from the masters (which must be saved for use next month).

When this first computer operation is completed, the master deck will be unchanged. The new detail deck will be the same as the original except that the price of each sale will be punched on each sales card. The product summary will be as shown in Figure 4.

Product Number	Total Sales
1190	868.32
1213	76.50
1655	200.00
1656	72.00
4192	153.71
4992	193.80
5062	139.65
	1703.98

Figure 4. Summary of sales by product in sequential file processing example.

Before proceeding with a description of the remainder of the processing (the summary of sales by salesman and district), we may investigate a way of presenting graphically the steps so far covered.

This may be done with a *work-flow chart*, or simply *flow chart*. A flow chart is a graphic representation of the complete system in which the input data is converted to final documents. In other words, a flow chart shows *what* the major processing steps are, without detailing *how* they are done; the latter is the concern of a *block diagram*, to be discussed a little later.

A flow chart uses lines and arrows to connect symbols that stand for documents and operations. Some of the standard symbols are shown in Figure 5. They are most easily drawn with the IBM Charting and Diagramming Template. A *source document* is any representation of information that becomes input to a data processing operation. In our example, the only source documents are the sales reports. The symbol shown for a file applies only to a card file, of course; the file concept is broader than its card implementation, as we shall see.

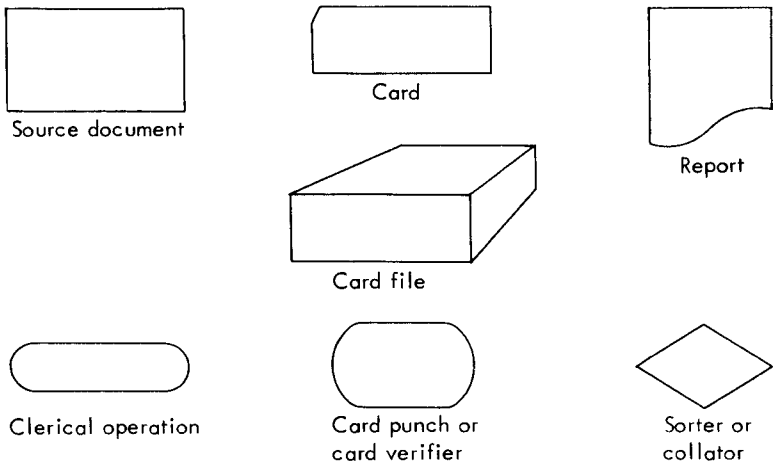

Figure 5. Some of the standard flow-charting symbols and their meanings.

The *card punch* is naturally a device for punching holes in cards. In our example it happens that only numbers are punched; most card punches can also punch letters and certain other symbols. The card verifier looks about like the card punch, but has no mechanism for punching holes. Punched cards are run through the verifier, with the verifier operator pressing the keys in the same way (hopefully) as the punch operator did. When a key on the verifier is pressed, the equipment checks that the key corresponds to the hole punched in that column. If the entire card proves to be correct, it is notched at the end to show that it has been verified. If the card is wrong or the verifier operator makes a mistake, a red light is turned on. The verifier operator can now try again; if the card is actually wrong, it is notched at the top to show that it must be repunched. It is of course possible that the two operators may make the same mistake, but this is sufficiently unlikely to make the technique acceptable in most situations.

With a flow chart we can represent the operations in our example that have been described so far, as shown in Figure 6. It may be seen that the pictorial representation is much easier to follow than the verbal description, once the basic concepts are understood.

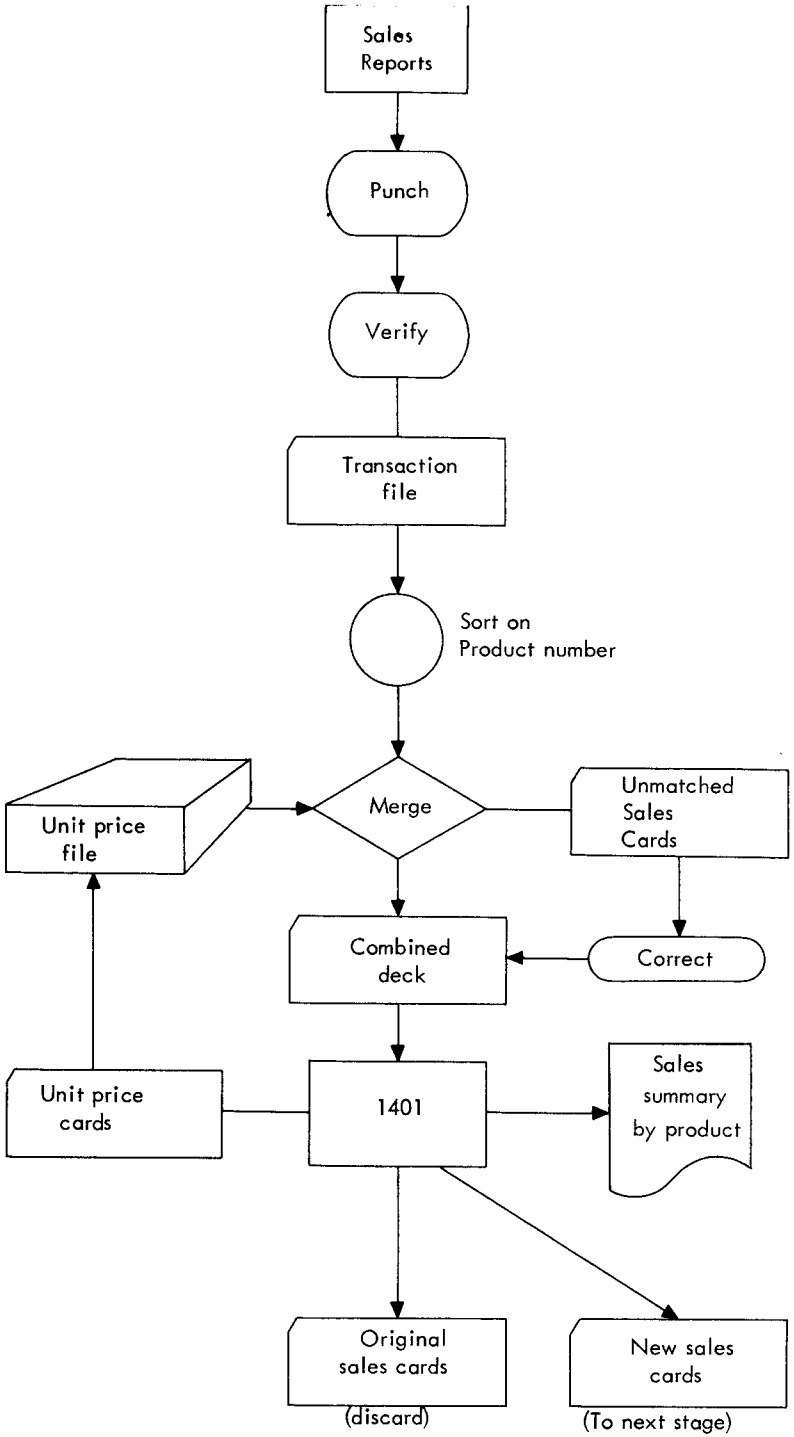

Figure 6. Flow chart of sequential file processing example, through the sales summary by product.

With the simplification made possible by the tool of the flow chart, the rest of the procedure is much easier to describe. We are required to produce a summary of sales by district and salesman. To do so, the new details must be resorted, so that all the cards for one district are together; within each district, all the cards for each salesman must be together. After sorting, another computer run produces the summary required. As the cards are read by the computer, the sales total for the month is computed again. This figure should obviously be the same as the total at the end of the previous summary, to give a check on the correctness of the processing. A number such as this is often called a *control total;* the term is also used to denote a total that has no other purpose than that of checking accuracy.

The flow chart of this part of the processing is shown in Figure 7. The new details, after they have been sorted by salesman and district, appear in Figure 8, and Figure 9 is the sales summary by salesman and district.

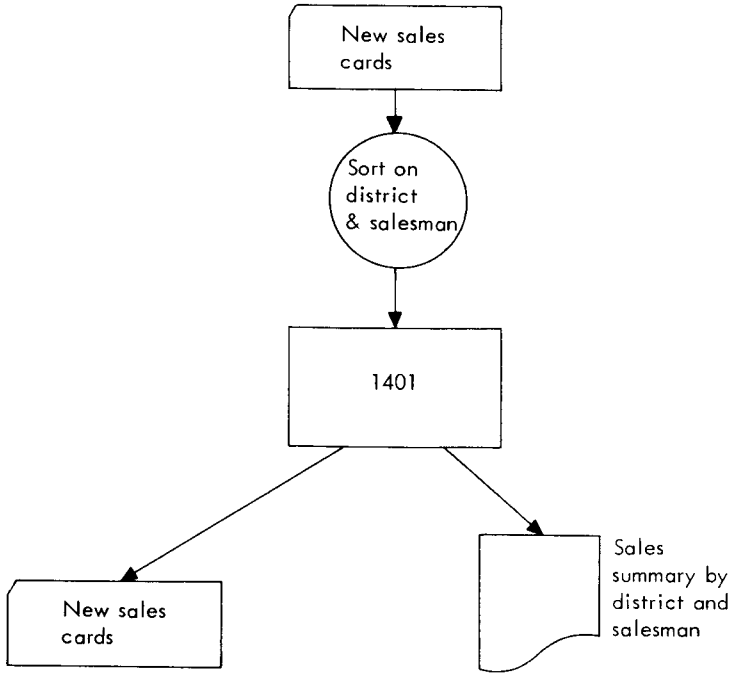

Figure 7. Flow chart of sequential file processing example, from new sales cards to sales summary by district and salesman.

Product Number	Quantity	Salesman	District	Sales Price
1190	30	10	1	129.60
4992	11	10	1	112.20
1213	2	20	1	51.00
4192	12	20	1	97.08
1655	20	6	2	16.00
1656	4	6	2	72.00
5062	20	6	2	29.40
1213	1	41	2	25.50
1190	100	41	2	432.00
1655	80	31	3	64.00
4992	8	31	3	81.60
1190	55	32	3	237.60
4192	7	32	3	56.63
5062	75	32	3	110.25
1190	16	61	3	69.12
1655	150	61	3	120.00

Figure 8. Sorted new details for sequential file processing example.

Salesman		District		Total
10	241.80			
20	148.08			
		1	389.88	
6	117.40			
41	457.50			
		2	574.90	
31	145.60			
32	404.48			
61	189.12			
		3	739.20	
				1703.98

Figure 9. Summary by district and salesman.

This discussion has said nothing about how the various computer operations are to be carried out. We are still not prepared to go into the details of this matter, but we can at least consider the overall picture of what the major computer operations are and their sequence. For this purpose it is convenient to employ a *block diagram*, which is considerably more detailed than a flow chart. A flow chart shows only the major steps in the processing as the work "flows" from machine to

machine. A block diagram, on the other hand, shows *how* the task of each machine is accomplished. In this book our primary concern is the computer, and block diagrams will be used to show the sequence of data movements, computation, and decisions in which the computer is involved.

The symbols used in flow charting are also used in block diagramming, but with different meanings. Figure 10 shows the meanings attached to the symbols needed in this example.

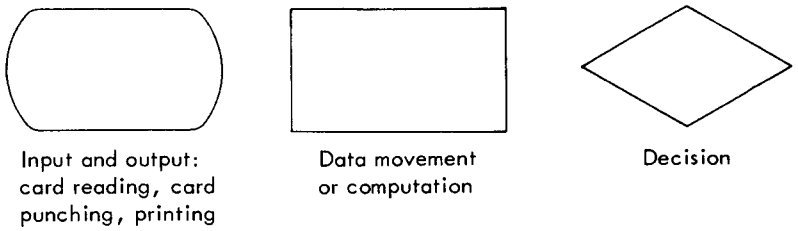

Figure 10. Some of the standard block-diagramming symbols and their meanings.

Figure 11 is a block diagram of the computer operation in the second part of our example, the summarization by month, district and salesman. The first step is to read a card. This means that the information on the card is copied into the storage of the computer, where it is available for later operations. With the data in storage, some of it is next copied to other places in storage so that it will be available after the data from another card has been copied into the storage areas occupied by the data from the first card. These data transfers are shown in three different boxes in the block diagram, because some of them are used at different times later. An arrow, in this connection, means "goes to."

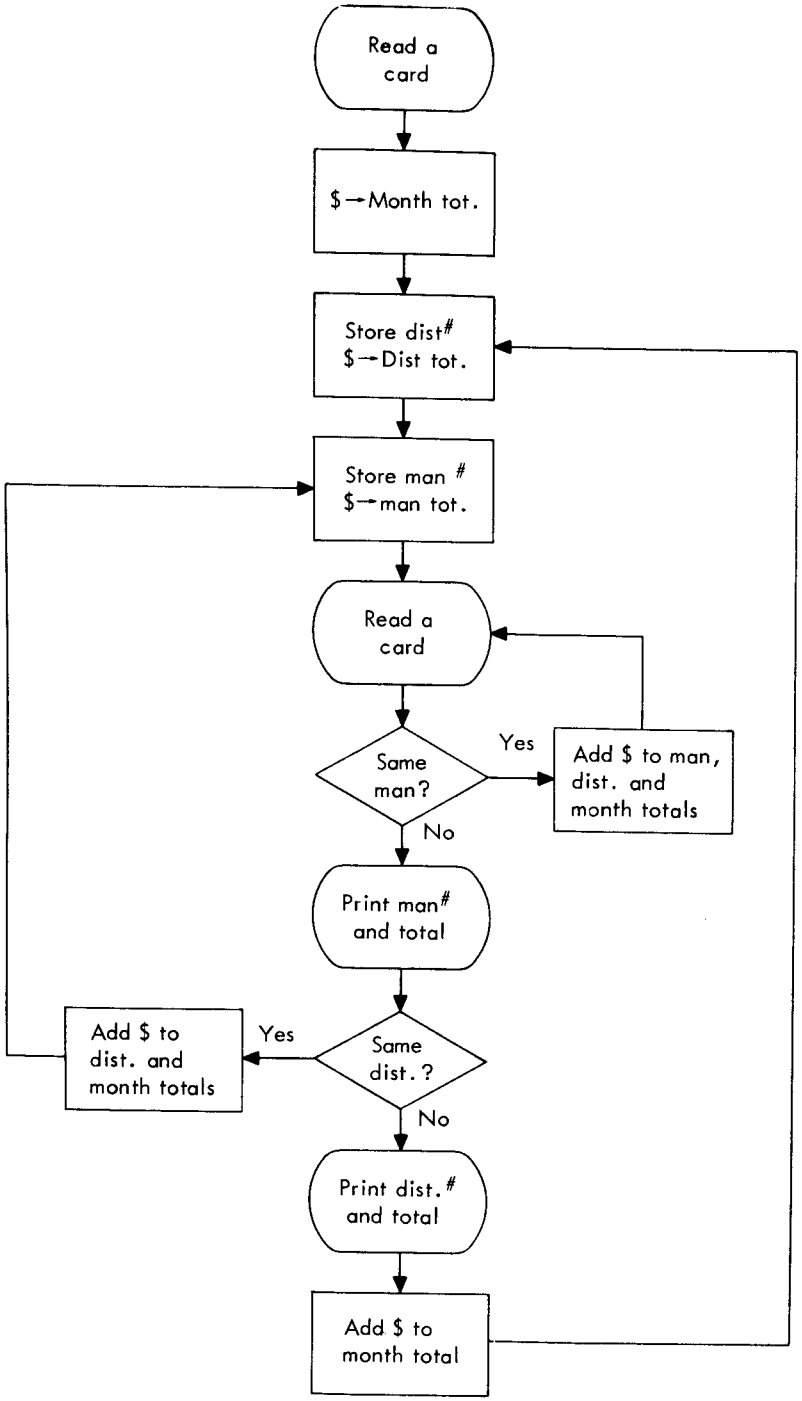

Figure 11. Block diagram of the computer run to produce a sales summary by district and salesman.

When the first card has been read and the data moved, another card is read. Now a decision is necessary: Does this card belong to the same salesman? If it does, then the price of the sale should be added to each of the three *accumulators* that will develop the three totals. If it refers to a different salesman, then the number and total of the previous man should be printed. Next a check must be made to see whether this card refers to the same district. If it does, then the sales price should be added to the district and month totals, the salesman number stored as the one against which to compare the next card, and the sales price stored in the accumulation for the man. This last will destroy the previous salesman total, which has now been printed. Next another card is read and the process repeated. If the second test indicates a new district, then the district total must be printed, and the other operations carried out as shown.

The student is strongly urged to follow through this block diagram in detail, using the data presented previously. The test required to detect the last card of the deck is not shown here; exercise 3 considers this problem.

This example brings out a number of important concepts, which may be summarized as follows:

1. The unit record concept. A punched card has the important advantage that it moves as a unit; this is not necessarily true of other storage media. When the key is used to control sorting or collating, all the other information on the card moves with the key. For this reason the conventional IBM card equipment is often referred to as *unit record* equipment. (And, of course, this example can be done entirely with unit record equipment; in fact, it is a typical application.)

2. Control levels. The salesman number controls one level of totals, and the district number a higher level that includes the salesman totals. The total by salesman is called a minor total, and the total by district a major total. There could also be one or more intermediate levels of control, as for instance if each district had branches out of which the salesmen worked.

3. Sequential file processing. When the files to be processed can be processed only sequentially, it is necessary to arrange all of the files into the same sequence before processing. This becomes the basic consideration in organizing the processing.

4. Batch processing. Since it is necessary to read the entire master file, including those records not affected, in order to process even a few transaction records, it is necessary to save the transactions until a *batch* of them has been accumulated. In many applications, as in the example above, this is a natural mode of operation; in others it is a decided disadvantage, and we turn to the *random access* file storage methods.

Review Questions

1. *Describe how you would look up a telephone number if the directory had to be "processed" sequentially. What is the "key"?*
2. *Why would it not be feasible to have several sales reports punched on each card of the transaction deck?*
3. *In the example, there can be several details with the same product number. Could there ever be several masters with the same product number?*
4. *What would happen in the merging operation if the last card of the sorted transaction deck were inadvertently placed at the beginning of the deck?*
5. *In the sample data for this example there was a mispunched sales card, which was detected because there was no master card corresponding to the incorrect product number. Suppose the erroneous product number had been the same as some master card product number; would the error have been detected?*
6. *Suppose that in the various card handling operations between the two summarization runs, one sales card got lost. What would signal the error?*
7. *If a salesman could report sales through more than one district, could the salesman and district summaries be produced in one summarization run?*

1.4 An Example of Random Access File Processing

The outstanding feature of sequential file processing is that the *entire* file must be read each time *any* transactions are processed against it. This in turn forces us to sort the transactions so that the master file need be read only once. Furthermore, it is necessary to accumulate the transactions into batches of fair size before doing any processing, in order to reduce the number of times that the entire file must be read.

In many cases the nature of the job is such that these factors are not actually restrictions. For instance, in the example of the last section, it is completely natural to accumulate a month's sales cards before running the monthly summary. The effort of sorting is more than compensated by the economy of sequential file storage media.

In other situations, however, the application demands that records be kept on a current basis, or that the information be more readily available than it usually is with a sequential file. In such cases the random access file becomes necessary.

For an illustration, let us consider the same problem as in the last section, but with two additional requirements. Besides preparing

monthly sales statistics, we are required to keep records on the inventory of each product, and to be able to answer on short notice several kinds of inquiries on inventory status and sales position.

As before, there is a master file consisting of one record for each product, but now the records contain more information. Besides the product number and the unit price, each record contains the number of units available for sale and the accumulated sales amount for the month. There is also a separate record for each salesman and each district, showing sales for the current month.

As the sales information is received at the data processing center, cards are punched and processed against the master file immediately, perhaps as often as several times a day. The processing of a sales card now consists of the following steps:

1. Locate the master record for the product number.
2. Determine whether there is enough of the item in stock to be able to fill the order. If so, proceed with the processing; if not, write an out-of-stock notice.
3. Subtract the number of units sold from the number of units in stock.
4. Extend the price—that is, multiply number of units by unit price.
5. Add the total sales price to the accumulated sales of the product for the month, which is now also in the master record.
6. Write the modified master record back in the file.
7. Locate the record for the salesman who sold the order, add the total sale price to the total of his sales for the month, and write the updated record back in the file.
8. Do the same for the district in which the salesman works.

This procedure is shown in the block diagram of Figure 12. The only new technique here is the location of the proper master record. How this is done depends on the physical device used to store the master file, a subject to which we shall return in Section 9.

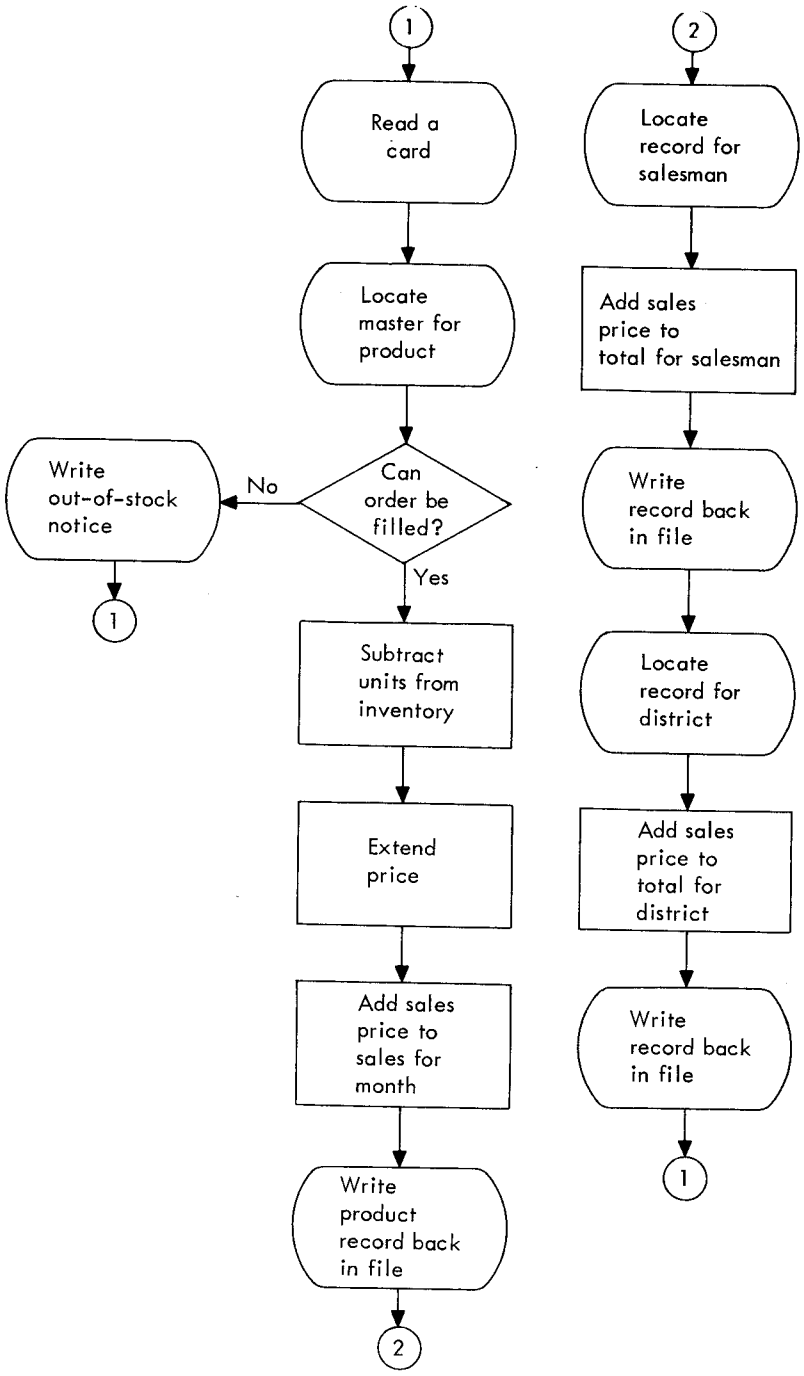

Figure 12. Block diagram of the computer operations in updating a sales statistics file, using a random access storage device.

So far we have considered only the handling of the sales cards. The complete system must obviously be able to do considerably more. If it is to keep inventory records, there must clearly be some way to enter information on the addition to stock of items manufactured by the company or received from suppliers. This can easily be done by setting up a card rather similar to a sales card—that is, containing a product number and a number of units—but which also contains a code that will be recognized by the computer program as indicating an addition to stock rather than a sale.

There must also be a procedure for producing sales summaries, as before. Besides the monthly summary, however, it is now possible to provide a summary on demand, at any time during the month. This would not ordinarily be a complete report, but only the statistics on selected items. Such spot summaries could be requested by other specially coded cards similar to the sales card, showing the product, salesman or district for which a summary is desired.

It goes almost without saying that this is a considerably oversimplified example. The inventory portion, in particular, does not take into account many factors that would be required in any actual application.

Comparing this procedure with the sequential file processing example, the following characteristics stand out:

1. Sales reports can be processed as quickly as they are received, rather than as batches are accumulated, since only the correct master records need be read. The master file information is thus always up to date.
2. Information from the master file is available on demand, with little delay.
3. No sorting of the transactions is required.

Why then are random access files not universally used? The simple answer is that the same capacity costs more in a random access storage medium than in a sequential access medium. Furthermore, many applications have little need for hourly or daily availability of the latest information; for them, a random access file represents a pointless expense.

In short, when immediate access to master file information is not needed, a sequential file is perfectly adequate and is less expensive. When immediate access is essential, a random access file provides conveniences that more than compensate for the additional cost.

Review Questions

1. *What is the most important characteristic of a random access file?*
2. *Would it introduce any complications in this example if a salesman could work out of more than one district?*
3. *In this example we did not mention any error checking. This is partly because there are fewer card handling steps where errors could be made, but at least one of the error checks in the sequential file version can still be made. What is it?*
4. *At the end of each month a sales summary for the month would be produced. After doing so, what should be done to the file information to prepare for accumulating the next month's statistics?*

Exercises*

**Answers to starred exercises appear in Section 12.*

*1. In the block diagram of Figure 11, insert the additional operations necessary to produce a count of the number of sales made by each salesman.

2. In the block diagram of Figure 11, insert the operation necessary to sequence-check the detail cards on district number—that is, determine that no district number is smaller than the previous one.

*3. Suppose that after reading the last card of the deck an indicator is automatically turned on. Insert in the block diagram of Figure 11 the operation necessary to test such an indicator and to use the result of the test to wrap up the processing when the last card has been read. This will require printing of all three totals, including now the month total.

4. Draw a block diagram of the computer operations in the summarization by product, in the sequential file version. Assume that there is some simple way to distinguish between a master and a detail—for example, a decision box which asks "master or detail?" following the reading of a card.

Suggestions: Draw the block diagram first without any consideration of how to start or end the process. This will be the heart of the run: obtaining the unit price of each product, extending each detail, and getting the summary for each product. To do this much, it will be necessary to use the detection of a new master card to cause the printing of the summary for the previous product and to store the new unit price for extending the next details. Be sure that nothing is printed when two masters in succession are read, and that after printing each summary the total storage area is cleared.

After this part has been worked out, it should not be too difficult to add the operations necessary to start properly and to make use of a last card test for stopping.

*5. Suppose that the sales card in the sequential file in this example had only the salesman number and no district number. There is an auxiliary master file that gives the district in which each salesman works; this file is in salesman number sequence. Extend the flow chart of Figure 6 to include the steps necessary to punch new sales cards with district number.