Principles of Programming

Section 10: Planning and Installing a Computer Application

IBM Personal Study Program

Section 10: Planning and Installing a Computer Application

In this section we shall consider an inventory control problem that is a somewhat more realistic version of the case study at the end of Section 8. This case study will still not show the entire complexity of a normal inventory control job, but it is close enough to give a fair indication of the work that must be done in setting up a data processing application. We shall use this application as the framework for considering the various steps in going from a problem statement to a running computer program.

10.1 Problem Statement

A certain company employs about 1,000 workers in building small to medium electric motors. The company has an inventory of 20,000 stock items. Four hundred of these are motors and related items built to stock; the remainder are raw materials and subassemblies. The company also manufactures equipment to special order, but since it is never stocked this type of finished product is not included in the inventory control system.

Inventory records in the past have been kept on ledger cards. As the company's business has expanded, this method has become more and more cumbersome, expensive and time-consuming. The company has decided to obtain a 1401 tape system for this problem and for a variety of other work such as payroll, production scheduling, cost accounting and a small amount of engineering calculation. The company anticipates that keeping inventory control records with the computer will reduce costs slightly, provide more accurate inventory control information, reduce clerical delay, and eventually provide the basis for a more thorough management control of inventory position.

The master file on magnetic tape will contain the following information:

Part Number
Abbreviated Alphabetic Description
Quantity on Hand
Quantity on Order
Reorder Point (the point at which more stock is ordered)
Reorder Quantity (the size of the order that is placed)

Code Character (indicating whether this is a finished item for sale, or a raw material or subassembly)
Unit Price for Finished Goods
Year-to-Date Sales for Finished Goods

In this case study, there are four types of transactions against this master file:

1. Issues. These refer to stock that has been issued by the stockroom either to purchasers or to the manufacturing operation.
2. Receipts. These refer to inventory items received in the stockroom.
3. Orders. These are orders placed by purchasing or production control for materials, subassemblies or finished stock.
4. Adjustments in the quantity on hand. These are a result of such things as recounts, loss and spoilage.

The processing required may be summarized as follows. The inventory tape is to be updated daily with the transactions being processed against the master file in much the same way as in the case study of Section 8. Adjustments replace the quantity on hand in the master file record. Receipts are added to the master file quantity on hand. Issues are subtracted from the master file quantity on hand. Orders are added to the quantity on order in the master file. Since the system is being set up so that nothing is ever received in the stockroom that was not ordered either by purchasing or production control, receipts also represent a fulfillment of an order that was placed previously. Therefore, receipt quantities are subtracted from quantity on order.

Before an issue quantity is subtracted from quantity on hand, a test is made to determine whether there is enough stock on hand to supply the amount specified. If there is not, we reduce the quantity on hand to zero and print an out-of-stock notice. If the quantity on hand plus the quantity on order falls below the reorder point, we print a recommendation to purchasing or production control that more of this item be ordered. We do not add this quantity to the quantity on order until we subsequently receive an order transaction—that is, until purchasing or production control responds to the order recommendation.

For issues of finished goods, the transaction quantity is to be multiplied by the unit price and the product added to the year-to-date sales.

We may summarize how this application relates to the work of other departments within the company. The sales department sends to the data processing center notices of sales of standard items. These become issue transactions. Sales and engineering together determine the raw materials and subassemblies required to manufacture special orders and these, in turn, become issue transactions when these items go out to the manufacturing floor. When production control receives a notice that the level of a standard item has fallen below the reorder point, it is production control's responsibility to schedule the production of more

of this item. In doing so, the need will be created for raw materials from the stockroom, which once again become issue transactions. When items go into the stockroom after being completed in manufacturing, they become receipt transactions. When an order recommendation for raw materials goes to purchasing, a purchase order is normally immediately placed with a vendor, taking into account any special consideration such as the combination of orders, quantity discounts, etc. As soon as purchasing places the order, an order transaction goes to the data processor to indicate that the quantity has been ordered.

We see that virtually every part of the company is affected in one way or another by inventory control. One of the first things that would normally be considered in designing the data processing system would be to put much of the interdepartment communication into a series of related computer programs. This, however, would get us deeply in the area of systems design, which is beyond the scope of a book on programming.

10.2 Problem Analysis

The decisions described in the previous subsection fall in the category of what might be called overall systems design. There is still considerable systems-type work that must be done before programming can begin; we might call the following area detailed systems design. Both of these areas have a bearing on the choice of machine to be used and the precise machine configuration to be ordered. We are ignoring this subject entirely.

Detailed systems design in this application would include things like the following:

Card, Report, and File Formats. Decision must be made on the information formats to be used in the system. These decisions are made on the basis of such considerations as the following:

Human readability: Spacing of information within a line and of lines on a report.

Readability of cards after they have been punched.

Whether preprinted forms are to be used for output, or whether all headings are to be produced by the computer.

How many copies of each report are required for the various groups that need them.

Card-punching simplicity. Cards are often used as the original source document, where the user writes his order or receipt or issue notice on a card, from which the same card is punched. If this is to be done, it must be ascertained that the necessary portions of the card are visible when required on the card punch.

Whether any of the card or tape file formats must be compatible with other applications.

This entire area of forms design is a specialized one, best undertaken by someone with experience and training in it. The job is not as simple as it may appear, and an inexperienced person can create sizable difficulties with poorly designed forms.

Time Schedules. Considerable attention must be given to scheduling of computer time and data arrival. When is the computer available in relation to other work? What is the deadline on the arrival of input and what is done with late data? When must the reports be available to users? How do any peak loads such as at month or year end affect other applications being run on the computer?

Volume Data. What is the size of the master file in terms of number of records and number of characters per record? How many transactions of each type may be expected daily? Are there peak periods when the transactions pile up? Can the entire master file be contained on one reel of tape or must a multiple-reel file be set up?

Controls and Error Checking. How much and what type of error checking is to be done? Some things will be done as a matter of course: tape read checking, parity checking throughout the computer, and, in most well-run installations, label checking and record or block counts. Beyond these obvious and simple possibilities, there are many other things that can be done. These include:

1. Batch control totals on transactions.
2. Balancing equations of the general sort: Old Balance + Receipts − Issues = New Balance.
3. Testing for invalid conditions such as nonexistent classification codes, negative quantities on hand, impossible dates, etc.

The cost of checking for errors must be weighed against the cost of not checking for errors. It is clear that errors can cost money. What is sometimes overlooked is the fact that more money can be spent on error checking than it is worth. Unfortunately, no satisfactory formulas can be given for arriving at a decision, since it is usually very difficult to establish what the cost of an error is, or even to enumerate all the possibilities of errors. At the present state of the art of data processing, we can only suggest that experience is the best guide.

Master File Creation. At some point, it is necessary to create the master file. If the job has previously been done with manual methods, it will be necessary to punch cards from the present records. If the job has previously been done with punched card techniques, it may be possible to convert the present file to magnetic tape with a separate computer program written for the purpose. Often the data in the cards is rearranged and new information added.

In some applications this problem of file conversion can be a major undertaking all by itself, requiring not only a good deal of personnel and computer time but careful planning in the scheduling of the conversion. This latter is made necessary by the fact that the business has to go on running while the computer is put into operation. Often the master file is much too large to permit a complete shutdown of this area of the company's clerical operation while the file is converted. The usual technique is to prepare the master file as of a specific date and continue with the manual or punched card methods until the computer system is ready to go into operation. During the period between conversion of the file and the termination of the previous methods, all transactions are saved. When the system is finally in operation, the first step is to process all the accumulated transactions against the new master file. If the file is extremely large, it is necessary, furthermore, to make the conversion gradually so that for a certain period of time part of the file will be processed by manual methods and part by electronic methods. We shall return to this point later and see that a period of parallel operation is usually desirable in any case.

We do not have sufficient space in this text to provide enough background information about our assumed company to permit a realistic discussion of the basis of the decisions that must be made in this area of detailed systems design. We must instead be content with a statement of the following decisions:

Formats. The master file tape will consist of blocks of eight records of 54 characters each, assigned as follows:

Information	Number of Characters	Character Position Within Record
Part Number	7	1 - 7
Description	12	8 - 19
Quantity on Hand (QOH)	5	20 - 24
Quantity on Order (QOO)	5	25 - 29
Reorder Point (RP)	5	30 - 34
Reorder Quantity (RQ)	5	35 - 39
Unit Price	6	40 - 45
Year-to-Date Sales	8	46 - 53
Code: 0 = Raw Material or Subassembly		
1 = Finished Goods	1	54

The transaction cards will have the following format:

Information	Number of Characters	Character Position Within Record
Part Number	7	1 - 7
Transaction Code	1	8
1 = Adjustment		
2 = Receipt		
3 = Order		
4 = Issue		
Transaction Quantity	5	9 - 13

An order recommendation card will be punched when the inventory falls below the reorder point.

Information	Number of Characters	Card Columns
Part Number	7	1 - 7
Code: 0 = Raw Material or Subassembly		
1 = Finished Goods	1	8
Recommended Quantity	5	9 - 13

Out-of-stock notices will be printed, with the format dictated only by normal readability requirements.

The printer will be used to print notices to the operator about the problem.

The effect of schedules on our planning of this program will be ignored since this subject gets us too deeply into the interrelationships in the company, interrelationships between different applications on the computer, and the detailed requirements placed on this application by management.

Volumes will similarly be largely ignored except to note that the master file is small enough to fit on one reel of tape and small enough to permit creation of the master file in one step. (These are poor assumptions in many actual applications.)

Programmed error checking will include certain checks based on the organization of the file, such as checking for nonexistent part numbers and invalid transaction codes. These are in addition to the label checking and tape checking that will be automatically handled by the Input Output Control System.

Before the transactions can be processed against the master file, they must be sorted into the same sequence as the master file. This will be done here using a standard tape-sorting package that is available for the 1401, as for most computers. The sorting program is available in generalized form, requiring only that the programmer supply a few items of information about the file and machine configuration in order to produce a sort program tailored to his needs. A generalized sorting routine of this type relieves each programmer of the large amount of effort required to write a specific sort for each job.

It is a good idea to analyze each application to determine whether tape sorting or card sorting is more effective. Tape sorting is considerably faster, but it does tie up the relatively expensive computer. Card sorting is sometimes much less expensive, if the control field is short, but the total job time is greater. This is a decision that must be made for each application.

10.3 Block Diagram and Program for Inventory Control Processing

A block diagram of the processing to carry out the operations described in the previous subsection is shown in Figure 1. The bulk of the file processing logic is the same as in the corresponding block diagram in Figure 7 of Section 8, but there are a few differences.

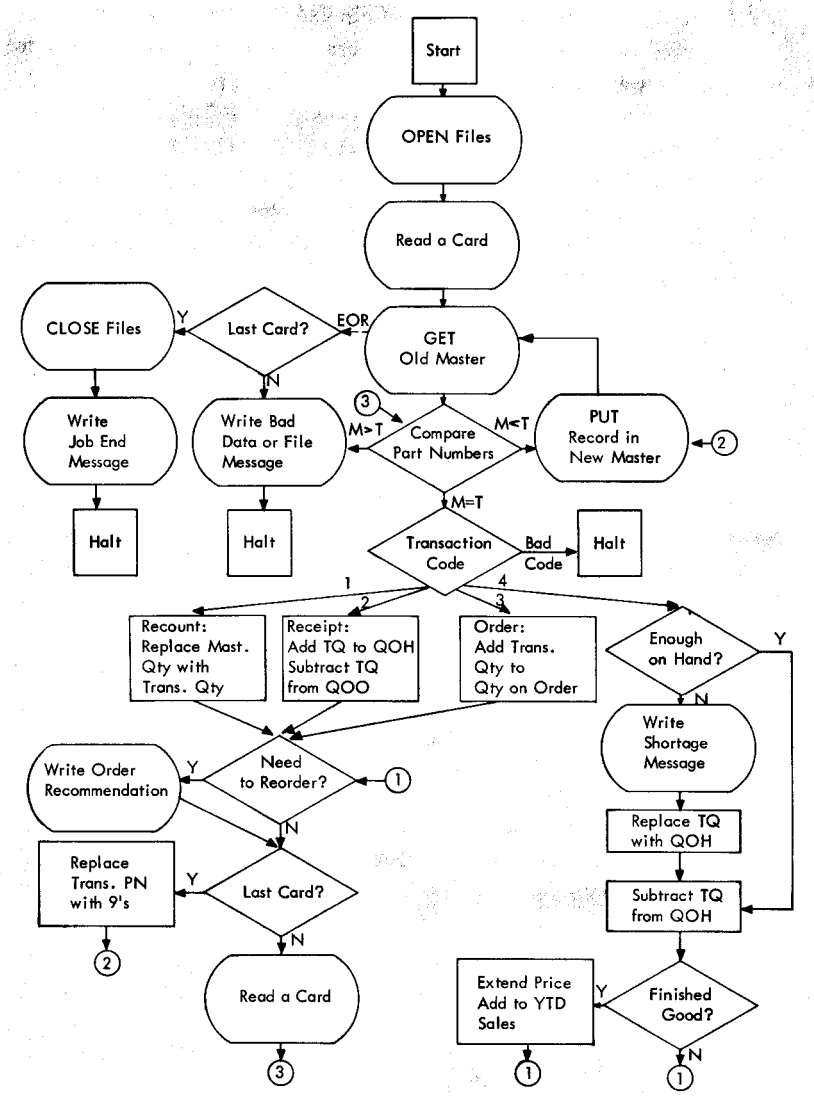

Figure 1. Block diagram of the solution to the inventory control application described in the text

In any file processing logic we must handle the situation where master file items remain after the end of the transactions has been reached. In the previous block diagram and program we set a switch for this purpose. Here we have followed an alternative procedure, which is equally good in most circumstances. When the last card is detected, we replace the transaction part number with 9999999, which is the largest possible "part number"; we assume that there is no *actual* part number 9999999. This will cause the comparison to show that the part number from the master record is less than the "part number" from the "card," which in turn will cause the old master records to be copied onto the new master tape.

The logic of the processing of the transactions is a little more complex now, because we are required to do more. Recounts are handled as before, by replacing the master quantity on hand with the transaction quantity. A receipt must be added to the quantity on hand, but also subtracted from the quantity on order—since we assume that nothing is ever received unless it has been ordered. (A situation not satisfying this assumption could be handled with a recount entry.) The transaction quantity of an order is simply added to the quantity on order.

Before, we merely subtracted an issue quantity from the quantity on hand. Now, we are required to determine that there is sufficient stock to fill the order before doing this. If the quantity on hand is not great enough to fill the issue request, we write a stock shortage notice, and replace the transaction quantity with the quantity on hand. This will assure that when the year-to-date sales are updated, we do not credit ourselves with selling more than we had. It also means that when the transaction quantity is subtracted from the quantity on hand, the quantity on hand will be reduced to zero, as it should be. (This could of course be handled in other ways, but this way will simplify coding.)

If the item represents finished goods that are being sold to a customer, which we can determine by inspecting the code in the master record, we update sales.

In any case, we next determine whether the quantity on hand plus the quantity on order has fallen to or below the reorder point, and punch an order recommendation if so. We do *not* immediately add the reorder quantity to the quantity on order; this will be done only when an order transaction is entered, indicating that purchasing or production control has responded to the recommendation.

Note the use of connectors (small circles enclosing numbers) to avoid long lines on the block diagram.

The program in Figure 2 follows the block diagram fairly closely. It provides a review of 1401 coding, with no new concepts being introduced.

FORM X24-1330-1
PRINTED IN U.S.A.

IBM

Program _____ 1401/1410 AUTOCODER CODING SHEET Identification 76 80

Programmed by _____ Page No. 1 2 of 4

Date _____

Line	Label	Operation	OPERAND
0.1	START	OPEN	OLDMST,,NEWMST
0.2		R	TRANSACTION CARD
0.3	READM	GET	OLDMST,,WORK
0.4	COMP	C	TRANPN,,MSTPN COMPARE PART NUMBERS
0.5		BH	ERROR LOW TRANS., SEQUENCE ERROR
0.6		BL	WRITNM LOW MASTER - NO ACTIVITY
0.7		BCE	RECOUN,,TCODE,1 TEST
0.8		BCE	RECPT,,TCODE,2 TRANSACTION
0.9		BCE	ORDER,,TCODE,3 CLASSIFICATION
1.0		BCE	ISSUE,,TCODE,4 CODE
1.1		LCA	MESSG1,,2,25 CODE, NO GOOD
1.2		W	
1.3		H	X-3
1.4	RECOUN	MCW	TRANQY,,MSTQOH
1.5		B	READ
1.6	RECPT	A	TRANQY,,MSTQOH
1.7		S	TRANQY,,MSTQOO
1.8		B	READ
1.9	ORDER	A	TRANQY,,MSTQOO
2.0		B	READ
2.1	ISSUE	C	TRANQY,,MSTQOH ENOUGH ON HAND Q
2.2		BL	SHORT NO
2.3		B	OK YES
2.4			
2.5			

Figure 2. Autocoder program of the procedure diagrammed in Figure 1

Line	Label	Operation	OPERAND
0,1	SHØRT	LCA	MESSG2,,249 WRITE
0,2		MCW	TRANPN,,210 STØCK
0,3		MCW	TRANQY,,223 SHØRTAGE
0,4		MCW	MSTQØH,,239 MESSAGE
0,5		W	
0,6		MCW	MSTQØH,TRANQY PUT QTY AVAIL IN TRANS
0,7	ØK	S	TRANQY,,MSTQØH
0,8		BCE	REØRD,,MCØDE,,0 BRANCH IF RAW MATL,SUBASSY
0,9		MCW	UNITPR,,MULT-6 FINISHED GØØD, -
1,0		M	TRANQY,,MULT EXTEND PRICE
1,1		A	MULT,,YTDS ADD TØ YEAR TØ DATE SALES
1,2	REØRD	ZA	MSTQØH,,TEMP COMPARE QTY ØN HAND PLUS
1,3		A	MSTQØØ,,TEMP QTY ØN ØRDER
1,4		C	TEMP,,RP WITH REØRDER PØINT
1,5		BL	LCTEST NØ REØRDER
1,6		MCW	TRANPN,,107 YES REØRDER - PUNCH PART
1,7		MCW	MCØDE,,108 NUMBER CØDE AND
1,8		MCW	RQ,,113 REØRDER
1,9		P	QUANTITY
2,0	LCTEST	BLC	LC LAST CARD Q
2,1		R	COMP READ A CARD AND BRANCH
2,2	LC	MCW	NINES,TRANPN FØRCE HIGH TRANS PART NØ
2,3	WRITNM	PUT	WØRK,,NEWMST
2,4		B	READM
2,5			

Figure 2 Continued

Line	Label	Operation	OPERAND
0,1	EØR	BLC	WRAPUP HERE FRØM GET RØUTINE
0,2	ERRØR	CS	0299 CLEAR STØRAGE
0,3		LCA	MESSG3,,229 WRITE
0,4		W	ERRØR MESSAGE
0,5		H	*-3
0,6	WRAPUP	CLØSE	ØLDMST,,NEWMST
0,7		CS	0299 CLEAR STØRAGE
0,8		LCA	MESSG4,,212 WRITE JØB
0,9		W	END MESSAGE
1,0		H	*-3 AND HALT
1,1		H	
1,2			

Figure 2 Continued

Page No. 4 of 4

Line	Label	Operation	OPERAND
01	MESSG1	DCW	@BAD CLASS CØDE JØB HALTED@
02	MESSG2	DCW	@ØNLY PART AVAILABLE REQUESTED@
03	MESSG3	DCW	@FILE ØR DATA ERRØR JØB HALTED@
04	MESSG4	DCW	@JØB FINISHED@
05	0001	DA	1X80
06	TRANPN		1,,7
07	TCØDE		8,8
08	TRANQY		9,,13
09	WØRK	DA	1X54
10	MSTPN		1,,7
11	DESC		8,,19
12	MSTQØH		20,,24
13	MSTQØØ		25,,29
14	RP		30,,34
15	RQ		35,,39
16	UNITPR		40,,45
17	YTDS		46,,53
18	MCØDE		54,,55
19	MULT	DCW	0000000000000
20	TEMP	DCW	00000
21	NINES	DCW	9999999
22		END	START
23			

Figure 2 Continued

10.4 Program Checkout

Any sizable data processing application provides hundreds of opportunities to make mistakes that will completely invalidate the program. These may range from simple slips of the pencil that result in nonexistent addresses, to block-diagramming errors that destroy the logic of the program, to misunderstandings of the intended procedures. A program is not finished until it has been thoroughly tested to remove all programming mistakes and to establish that the program properly carries out the intentions of the person who defined the problem. This process of detecting and correcting errors and proving the correctness of a program can easily take weeks.

There is no single way of solving all the problems of checkout. We must rely on a combination of procedures on the part of the programmer and on specialized checkout programs that are available to help him. Above all, the programmer must exercise good judgment in determining how to go about the checkout process. This is one of the primary areas in which experience is essential to a satisfactory result. We can only point out what some of the standard procedures and tool programs are and encourage the reader to get much practice in this area, if possible under the guidance of an experienced programmer.

Desk Checking. One practice that is most strongly recommended is to check all programs very thoroughly before they are assembled. The programmer should plan to spend as much as several days prechecking a major program. This should be a character-by-character check to make sure that every symbol and every operation code is correct, that O's and 0's are properly distinguished, that core storage information is never destroyed prematurely, that loops are counted correctly, that decimal points are properly handled, that all utility programs have been used in accordance with specifications, that timing problems are properly handled, etc., etc.

There seems to be an almost overwhelming temptation when a program is finished to put it on a machine to see whether it works. We cannot emphasize too strongly that all experience indicates that time spent in desk checking at this point will save much more time later. It can easily happen that half a dozen errors of an easily-found variety can waste weeks of time and cause several unnecessary reassemblies. If possible, it is an excellent idea to have someone else check the program, on the theory that the original programmer may overlook errors because he knows what the program *ought* to do, and does not check to see that the program as he wrote it *actually does* what it is supposed to do.

A listing of the program as punched is a valuable tool in desk checking, since it allows the programmer to check for errors in punching or in interpreting his handwriting at the same time that he is checking the program as he wrote it.

It is recommended that a copy of the listing of the assembled absolute program be obtained; this is called the post-listing in the SPS and Autocoder systems. This listing provides not only a diagnosis of certain errors that can be detected by the processor, but also a very useful document for all of the following stages of program checkout. This usefulness is based on the fact that the original symbolic program and the assembled absolute program are shown on the same piece of paper. This allows the programmer to correlate the program that is actually running in the machine with the way that he wrote it to begin with. Most programmers file their original coding sheets at this point and rarely refer to them again.

The basic idea of program checkout is to put the program in the machine with suitable test data and see whether it computes correct results. At the outset, this test data may be very simple; the idea is to see whether the program produces any answers at all. The most common experience when the program is first put on the machine is that it stops before producing any results. This can be caused by a variety of errors. If the difficulty causes some sort of error indication on the computer console, then it is fairly easy to get back to the source of the trouble, correct it and try again. If the problem runs nearly to completion and then hangs up, or if it does produce problem answers but they are incorrect, then we have the problem of determining where in a program of many hundreds of instructions something went wrong. Two powerful techniques come into play at this point.

The first is the use of a storage print, or, as it is more commonly called, a memory dump. This provides a printed listing of exactly what was in core storage at the time the program was stopped. In the 1401 and in most computer systems, the listing is printed with line identifications that make it simple to associate storage addresses with the printed contents. The memory dump allows the programmer to inspect the results of program modification, it provides a complete listing of all intermediate and final results computed up to the point of the dump, it shows exactly what test data is being used, and it allows the programmer to determine whether anything in storage has been modified that should not have been.

Usually only a part of storage is printed; there is seldom any need to see the *entire* contents. The starting and ending addresses of the regions to be printed are entered from the computer console or from the card reader.

The entire procedure is so simple, and the assistance it provides so valuable, that it is strongly recommended that a memory dump be obtained at every checkout session. The alternative practice, followed by some inexperienced programmers, of copying down from the console a few supposedly critical numbers cannot be defended; it is inaccurate, time-consuming and generally completely ineffective.

The second important checkout technique is to check out the program in sections by providing for the printing of intermediate results. This allows the source of errors to be narrowed down successively. Furthermore, it assists in proving the correctness of a program. The purpose of checkout, after all, is not only to locate and correct errors, but to guarantee that no undetected errors remain. The only way to do this is to verify every partial and final result against known test cases; printing out the partial or intermediate answers helps in this process.

The intermediate answers may be obtained with memory dumps taken at various intermediate points, or they may be printed by instructions inserted in the program for the purpose. These may be left in the program until a final assembly, at which time they are removed, or they may be made conditional on the setting of a sense switch and left in the program.

This brings up a point that should be kept in mind: checkout is such an important part of the whole process of getting a computer application into operation that it must be planned for in writing the program in the first place. The provision of instructions to print intermediate results is only one example of this sort of planning. Others that may apply in various situations: placing all results in one area of storage to simplify printing; avoiding "tricks" in coding that may be difficult to test; including comments that make it easy to correlate the program with the block diagram; etc.

It is occasionally desirable to use a technique called *tracing*, which provides a listing of the instructions and data as each instruction in a section of the program is executed. This technique should ordinarily be used only when all else fails. When used indiscriminately, it can waste large amounts of computer time and still not provide the needed information.

A variety of specialized checkout programs have been produced for most computers. These may do such things as providing printouts of selected sections of storage during the execution of the program, analyzing storage after the program has been run to print all areas that have changed since the program was loaded, inserting halt instructions in all unused sections of storage so that the program will stop if it reaches an unintended location, etc. The availability and operation of these programs vary considerably from one machine to the next.

The final check that should be made on any data processing program is called *pilot operation*. This consists of using a large quantity of actual transactions from a previous period. Use of actual transactions gives a better test for possibilities that may not have been considered in making up test cases. The results produced by the computer system can be checked against the results produced by the previous manual or unit record methods. And since the transactions have already been processed, there are no business pressures to get the results

out, pressures which would prevent careful analysis of the operation of the program.

One of the major secondary benefits of pilot operation is that it provides personnel in the department for which the work is being done an opportunity to see whether the program actually does what they intended. It is all too easy for the problem originator to say one thing and mean another, or for the programmer to misunderstand what is meant by terms in an area with which he is not familiar, or for either party to overlook special conditions. The program is not actually finished until it has been proved to produce precisely correct results using real problem data in volume.

10.5 Going into Operation

After a new program has been properly tested, there still remain some problems in going into full-scale operation. The transition from previous manual or punched card methods involves a sufficient number of people and enough changes in procedures that careful plans must be laid to insure that the changeover is smooth.

The first thing that must be normally done is to catch up with the transactions that have occurred since the new master file was created. This is also an excellent chance to give the program one further shakedown before it is required to do all the data processing itself. During this process, a fair number of errors in the master file will normally be discovered. These may be the result of incorrect conversion or they may represent errors that were in the previous file all along and had not been detected. Actually, this process of file cleanup ordinarily continues into the first few weeks or months of operation of the system.

A common practice is to continue the use of the previous manual or punched card system for a few weeks in parallel with the operation of the new electronic system. This parallel operation provides a backup in case the electronic system develops difficulties that require it to be taken out of operation for a while to correct. The parallel operation also provides an excellent test of the accuracy and adequacy of electronic processing, since it is a simple matter to compare the results produced by the two systems. The processing of the transactions accumulated since the creation of the master file also provides a means of checking the two systems against each other.

As we noted before, many master files are so large that they must be converted in segments, since an attempt to convert the entire master file at one time would result in such an accumulation of transactions that it would be difficult to catch up. One common way to effect such a partial conversion is to break the file into segments arbitrarily on the basis of the keys of the records. Another possibility is to convert the

records for which there are transactions as the transactions arise. Any such partial conversion scheme must obviously be carefully planned in advance so that there is a minimum of confusion between the data processing center and those responsible for the previous methods. The difficulty, of course, is that the people who have been using the old methods will in most cases still be heavily involved in the new system. If the transition is not properly carried out, these clerical personnel may be overloaded by having to deal with two systems at once.

It is perhaps obvious that the introduction of various applications to be done with the computer should be spaced out. Any attempt to put several major applications in operation at one time would create peak loads both at the computer and in the rest of the organization that could very well cause failure of the whole system.

10.6 Documentation

A computer system is next to worthless if it is not adequately documented. To provide all of the information that is required for the many people in various parts of the company organization that must use the system, several different types of documentation are needed.

One type of documentation that is obviously necessary is a complete writeup of the program itself. This document, which is often called a *run manual*, normally contains some or all of the following information:

1. A run number and title.
2. The name of the programmer.
3. The date of completion of the program and of the last modification.
4. A sheet summarizing the computer operating instructions for ready reference, including labels and descriptions of tapes and their disposition, error or special procedures, rerun instructions, average run time and switch settings.
5. A one- or two-paragraph description of the purpose of the run.
6. A complete set of flow charts and block diagrams.
7. Completed forms where applicable, for: storage allocation; record designs; operating instructions for any off-line equipment.
8. An assembly listing of the program. Changes in the coding after the program has been checked out should be entered in red pencil, initialed and dated.
9. A sample of each report produced by the program.
10. Suggestions for future changes and warnings about making changes.

A document of this sort is obviously necessary if intelligent use is to be made of the program and if modifications and revisions in the program are to be made with minimum effort.

The programmer should prepare instructions for the computer operator covering machine setup, console switch settings, tape units required, carriage control tapes, error correction procedures, etc. This, of course, duplicates some of the material in the run book, but it should be remembered that the run book is too big a document to be kept at the computer console. Operating instructions for all programs are generally kept in a single notebook at the console.

A third type of document might be called a procedures manual. This is used by the people who make use of the computer, that is, the personnel in other departments of the company who originate data and use the results. Typical contents of such a manual are:

1. Exhibits of input documents, with instructions for their preparation and transmittal.
2. Exhibits of output forms and reports and an explanation of their contents, discussion of the frequency of preparation, etc.
3. Timing schedules for data submission and receipt of reports.
4. Handling of special circumstances.

10.7 Summary

Putting a data processing application on a computer involves a number of steps, carried out at various levels of the organization by many different people. It begins with a study of what the data processing needs are and of alternative ways of solving them. After it has been established that a particular computer is to be ordered, much work must be done in deciding just how to go about splitting up the company's data processing requirements into manageable pieces that can be set up on the computer. After this has been done, the general characteristics of each computer run must be planned, including file and record formats. Only at this point is it possible to write computer instructions. When instructions have been written, the accuracy of the program must be verified. The file conversion and the start-up of the application both require considerable planning. To this list should be added such activities as planning for the physical installation of the computer, the training of the people to program and operate it, indoctrination of the people in other departments of the company who will make use of the services of the computer, and an education program to introduce the computer to the entire company in a way that will minimize the ever-present fears about job security.

It must be admitted that in this complete list the subject of coding that we have discussed in this book is only one part, representing less than a majority of the time required to get into operation. The person who expects to be working closely with computers nevertheless needs to start his education with the subject of computer coding or programming, because without this knowledge a proper grasp of the more advanced subjects cannot be gained. Still, it should be realized that the area that we have introduced in this book *is* only the beginning and that the person who expects to be a truly professional computer expert has a number of years of apprenticeship and study before him.

Exercises

1. The block diagram and program of this section contain a questionable procedure: the reorder calculation is made after every transaction. If there are several issues for one part, it could easily happen that we will punch many order recommendations, which is at least pointless and perhaps confusing. Modify the block diagram so that the reorder calculation is made only when all transactions for a part number have been processed.

2. You are given a master payroll tape for a payroll of 4,000 hourly workers. Each record contains a payroll number in positions 1-5, an hourly pay rate of the form X.XXX in positions 22-25, and other information totaling 200 characters. You are also given a cost accounting tape containing one record of 80 characters for each of 800 active jobs in the company. Each record contains a job number in 6-9, a total-to-date cost in dollars in 49-54, and other information.

Each week there are about 40,000 labor voucher cards giving payroll number in 1-5, hours worked to tenths of an hour in 6-8, and a job number in 9-12.

You are required to prepare a weekly labor distribution report showing, for each job on which work was performed this week, the total direct labor cost for the week; the cost accounting tape must be updated to include this week's direct labor cost; for each man, a card must be punched showing gross pay for the week.

a. Draw a flow chart of the computer and punched card operations necessary to satisfy these requirements.

b. Draw a block diagram of the two computer runs required to get gross pay and to produce the labor distribution report.

c. Write programs corresponding to the block diagrams. State any additional assumptions that must be made to carry out these operations.