A View of Computers of 1952

Peter Naur
Datalogisk Institut, Københavns Universitet

Commentary 1997

The following text is a translation of a report in Danish titled: Beretning om programstyrede elektronregnemaskiner i England, U.S.A., og Sverige, dated May 1954. Otherwise it speaks for itself.

As to the effect of the views presented in this report it may be stated briefly that the most prominent Danish computer development during the years 1955 to 1960, conducted at Regnecentralen, was centered around the computer DASK, which was a modified version of the computer BESK described in the report, with program organization designed along the lines advocated in section 6 of the report.

Report on program controlled electronic computers in England, U. S. A., and Sweden

With a proposal for a Danish nomenclature

Peter Naur
Assistant at Copenhagen Observatory

May 1954

Contents:

Preface

The information in the present report is primarily based on a number of visits to English, American, and Swedish computer laboratories, that I have been able to make with the support of Ole Rømer Fondet and the Carlsberg Foundation from 1950 to 1953. Besides factual information certain personal views are presented. These to a large extent have been derived from the experience of the laboratories, whose form of operation I have had the opportunity of experiencing at first hand. Under other circumstances I might perhaps have arrived at different conclusions. In this connection it should be noted that I have had the opportunity of actually working only with one machine, namely the Edsac in Cambridge, England. Especially my view of the most proper organization of the work with an electronic computer has been strongly influenced by the points of view developed at Cambridge. As a whole the report has to be seen on the background of the possibilities of an electronic computer for use in Danish science.

The last section of the report gives an account of the Danish nomenclature adopted.

Survey of visits to English, American, and Swedish laboratories

First stay in England: Worked with the Edsac, University Mathematical Laboratory, Cambridge, February to June 1951.

Stay in U. S. A., April 1952 to May 1953 (italicized periods were devoted mainly to studies of electronic computers):

1952 April 7, arrival in New York.

April 9, arrival at Yerkes Observatory, Wisconsin.

April 27 - May 15, Course in the use of the machine Univac, given by Remington Rand Corp., New York City, with visit to the laboratories in Philadelphia. Saw the machine SSEC, New York City. Visit to IBM Watson Laboratory. Princeton, New Jersey: Saw the machine at the Institute of Advanced Study. Washington, D.C.: Visit to The National Bureau of Standards where the machine Seac is to be found.

May 15 - July, stay at Yerkes Observatory.

July 10 - October 25, stay in New York. Attended a three week course with final examination in the use of IBM punched card machines at the Watson Laboratory. This course covered both electromechanical machines (type 602-A, etc.) and the electronic type 604 and CPC (The Card Programmed Calculator). Subsequently the machines of the laboratory, primarily the electronic 604, were used to solve differential equations of problems of the interior structure of stars.

1952 October 11 - 14, Visit to Cambridge, Massachusetts. Saw the Harvard Computing Laboratory with the machines Mark I and Mark IV.

1952 October 25 - 1953 February 7, Stay at Yerkes Observatory.

1953 February 10 - March 25, Stay at McDonald Observatory, Texas.

March 27 - 29, Stay at Lowell Observatory, Arizona.

March 30 - April 12, Stay in Los Angeles, California. Saw the electronic machine Swac at The Bureau of Standards, Los Angeles. Visit to Computer Research Corporation, Hawthorne, where the machine Cadac 102-A is being built. Visit to Consolidated Engineering corporation, Pasadena, where an electronic machine Model 30-201 is under development.

April 13 - 19, Stay at Berkeley, California. Saw the still incomplete machine Caldic at the University of California.

April 20 - 23, Stay at Lick Observatory, California.

April 27 - May 10, Stay at Yerkes Observatory.

May 11 - 17, Stay at Washington, D. C.

May 18 - 25, Journey to England.

Second stay in England:

May 26 - August 12, Carried out the calculation of the motion of planet 51 Nemausa through the period 1858 - 1960 with the aid of the Edsac, Cambridge. Saw the machine Elliott 401, on loan to The Mathematical Laboratory.

November 20 - 25, Visit to Stockholm. Saw the machine Besk.

General description of the program controlled electronic computers

The present report is concerned with a comparison of various possibilities in the practical design of an electronic computer. It will be understandable only to a reader who is familiar with the general properties of these devices. In this introductory section these properties will be summarized briefly. This also gives the opportunity to introduce a Danish nomenclature, which will be used in what follows.

An electronic computer is an apparatus which is able to carry out extensive series of calculation processes automatically, with the aid of electronic tubes. Such a machine must be equipped with five sorts of organs 1) An *input mechanism, i.e.* an apparatus that allows numbers to be fed into the machine from outside. 2) A *store* for numbers, *i.e.* an apparatus that makes it possible that numbers are held in the machine for an indefinite period of time. 3) An *arithmetic unit, i.e.* a system that is able to carry out arithmetic operations upon the numbers held in the store. The results of the operations will again be held in the store. 4) A *system of control* that makes it possible that any previously determined sequence of aritmetic operations will be performed by the machine. 5) An *output mechanism*, *i.e.* an apparatus which may transfer numbers from the store of the machine to a medium which is independent of the machine.

This general structure scheme allows a variety of practical realization. Such variety is also found in the earliest electronic machines, Eniac and SSEC. However, the machines whose construction has been started after 1946 are much more similar, since they have been designed around a coupling of the system of control and the store, which allows a very simple and effective structure. This coupling is based on the following consideration: Every computational problem is specified by two lots of information: 1) The numbers that enter into the problem, and 2) the information that determines the character of the operations. During the solution of the problem both of these lots have to be available to the machine. Looking at the work of the machine from this point of view it lies close to hand to suggest that both lots be held in the same store in the machine. Then the system of control continuously has to fetch information from the store, and this latter has to be regarded as an integral part of the system of control. This arrangement, which was proposed by von Neumann in 1946, is used in all large new machines. It gives rise to the designation *program controlled machines,* since the information in the store that specifies the operations is called *the program of the problem.*

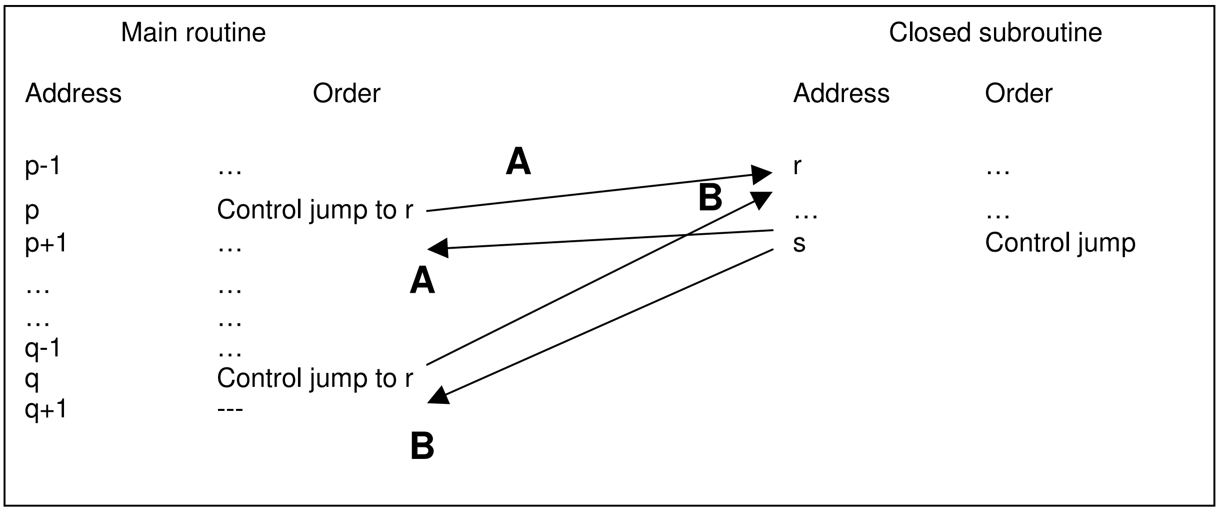
In a program controlled machine the significance of a definite number held in the store of the machine is not uniquely determined. It may, in fact, be interpreted by the machine both as a number and an operation. In such a machine the setting up the machine for performing a certain sequence of operations is equivalent to placing certain numbers in the store of the machine. These numbers will not have to be treated as such by the machine, but will have to be analyzed according to the code that transforms numbers into operations. The essential advantages hereby may be emphasized:

a) The information about what operations are to be performed is transmitted to the machine along the same channel, namely the input mechanism, as the numbers upon which the machine has to operate. Preparing the machine for a particular task thus ceases to be a special problem.

b) The operations may be modified by arithmetic operations. Since the store has only a finite capacity this increases the logical possibilities. It turns out that many practical savings may be obtained by the use of such modifications.

The machines that are described in the present report are all program controlled. We shall presently introduce a series of designations for functions and concepts that apply to such machines. The store from which the control system fetches its information is called *the working store*. The capacity of the working store is divided into *locations*, which are numbered. The numbers of the locations are called *addresses.* Each location will hold a *word.* Each word has a definite number of *digits* (decimal or binary). During the solution of a problem a certain part of the locations of the store are used to hold the numbers of the problem, while another part contains the *program.* The program consists of *elementary orders.* During the work of the machine these are executed one by one. The sequence is normally the one in which the orders have been placed in the store, such that the order held at the address *n+1* is executed after the one at address *n.* However, certain particular orders break this sequence and make the machine make a *jump* (*control jump*) from one address to an entirely different one. The control jumps are made first and foremost in response to *conditional orders.* With these orders the jump is made only if one of two conditions is satisfied, otherwise the control continues in the usual manner.

The program of a complicated problem usually consists of many separate parts that are performed by the machine in a definite sequence. These shorter parts are called *routines*. Within each routine there may again occur repeated applications of short or long sequences of orders. Short sequences of orders that are executed many times are called *loops*. A commonly used manner of combining a larger number of routines to a more comprehensive program is to work out the routines in the form of so-called *closed subroutines.* The work of the closed subroutines is subordinated to a *main routine* in the following manner: During the execution of the orders of the main routine the machine encounters an unconditional control jump, for example at location no. *p*, which transfers the control to the first order in a subroutine (the subroutine is *called*). Then the subroutine performs its work and ends with a control jump back to location no. *p+1* in the main routine. Schematically the use of a closed subroutine looks as follows:



The special advantages of using closed subroutines depend on a system of orders which allows that a subroutine is called from an arbitrary point in the main routine, and yet returns to that point after completing its work (thus the two jumps in the figure marked A, and the two marked B, belong together). One may therefore call a subroutine from several different points in a main routine. In any case the subroutine will return control to the order following the point where it was called.

The working store is usually of rather limited capacity, since it is required to work fast and therefore is costly. In many machines one has extended the accessible storage capacity by adding *auxiliary stores* of large capacity, but longer access time. The auxiliary stores are then subject to the control system of the machine, which in response to suitable orders will make a transfer to the working store of a part of the information in the auxiliary store. This information will be available for the ordinary processing in the machine only after having in this way been transferred to the working store. The information held in the auxiliary store may be either numbers and orders. An auxiliary store will therefore make it possible that problems involving both extensive numerical information and complicated programs are solved in one session with the machine.

The data delivered to the machine, both the program and numbers, have to pass through one of the input mechanisms. Likewise the results of the work of the machine have to be delivered to the environment through an output mechanism. Both of these parts of the machine will normally be subordinate to the control system of the machine. The entire work of the solution of the problems can thus be performed automatically by the machine.

Survey of the machines under review

In this section a survey of the machines that have been taken into account in the present report will be given. It is convenient already at this stage to divide them into groups:

A. *Large machines* having *fast stores* built only in one copy, often essentially as *experimental models.* These machines in most cases are continuously subject to extensions and improvements, and it is difficult to establish a definite date of their completion. Although most of them were intended as objects of study they are all of them being used to produce useful work. The group includes the following machines:

Edsac, Cambridge, England.
The machine of the Institute of Advanced Study, Princeton, New Jersey.
Seac, Washington, D. C.
Harvard Mark IV, Cambridge, Massachusetts.
Whirlwind I, Cambridge, Massachusetts.
Swac, Los Angeles, California.
Ordvac, built in Urbana, Illinois.
Illiac, Urbana, Illinois.
Besk, Stockholm, Sweden.

B. *Large, expensive, commercial* machines. The machines in this group have been built in several copies with a view to applications to administrative and industrial problems. Their expensiveness is closely related to the speed of their internal store for numbers. They may also be characterized as the commercial machines having mercury delay line store or Williams tube store. The group includes:
The Ferranti Computer, built in Manchester, England.
Univac, built in Philadelphia, Pennsylvania.
IBM type 701, built in Endicott, New York.

C. *Smaller* machines, with store on *magnetic drum* or *magnetic disk.* The machines in this group do not differ from the machines in group B with respect to how complicated problems may be handled, since the capacity of the store is about the same. However, their speed of calculation is substantially lower. The group includes one completed commercial machine:
Cadac 102-A, built in Hawthorne, California.
One machine which was still being developed in April 1953:
Cec Model 30-201, Pasadena, California.
Further the machine:
Era 1101, Minneapolis, Minnesota,

and the still incomplete:
Caldic, Berkeley, California.

In what follows a comparative evaluation of the properties of these machines will be attempted.

Input and output mechanisms

The systems that are used to deliver data to and receive data from the electronic computers differ considerably from one machine to another. In evaluating the various systems it is convenient to group the media being employed in the following manner:

A. Media which may be used both for input and output:

Magnetic tape	may be overwritten
Magnetic wire	(*i.e.* data may be erased)
Punched cards	cannot be
Punched tape	overwritten

B. Pure input mechanisms:

Push buttons, including typewriter keyboard.
Telephone dials.
C. Pure output mechanisms:

Typewriters, teletype machines.
Cathode ray tubes.
Curve plotters.

The advantage of the media of category A is that results from the machine may be fed directly back into it again. For example the following machines have mechanisms for handling *magnetic tape*: Edsac, Seac, Harvard Mark IV, Whirlwind I, Univac, IBM 701, Cadac 102-A.

Seac in addition is equipped with mechanisms for *magnetic wire.*

Punched cards are used in the following machines: The machine of the Institute of Advanced Study, Swac, IBM 701, Cadac 102-A.

Punched tape is used in: Edsac, The machine of the Institute of Advanced Study, Seac, Whirlwind I, Ordvac, The Ferranti computer, Besk, Cadac 102-A, Era 1101.

The properties of these four media will now be discussed.

Punched tape and punched cards

These media are similar in many respects. They are both well known, well tested products, punched tape being used in teletype systems and punched cards used extensively for commercial calculations. For handling them there are in both cases standard usable systems, punching and printing mechanisms for the tapes, punchers, sorters, duplicators, and collators, for cards. Of the two the tapes are the simpler. The elementary mechanical reading process consists in reading one row of holes (*i.e.* a symbol corresponding to perhaps five 'hole'-'no hole' possibilities = 32 possibilities) and advancing the tape a stretch of one hole distance. In the mechanical reading of a card there is space for 12 × 80 = 960 'hole'-'no hole' possibilities, and it requires a system for advancing the next card to the right position. However, these extended demands on the mechanism for handling cards is counteracted by the commercial availability of such equipment. The use of cards gives the great advantage of making available simpler operations, such as rearrangements of the cards, with the aid of simpler machines, without putting a demand on the time of the main machine. Cards require considerably more space that tape, per digit they will hold, but is a much more convenient medium. The achievable speed of reading seems to be nearly the same for both media, since the fastest card reading mechanism (IBM's collator) reads 150 cards per minute = 150 × 80 decimals = 12 000 decimals per minute, while The Ferranti Computer and Besk are equipped with photoelectric and capacitive reading mechanisms for tape that read 200 × 60 = 12 000 five-hole rows per minute.[1] However, the cards are vastly superior in writing, since the usual card punches (such as IBM's reproducer) punch 100 cards = 8000 decimals per minute, while it appears to be difficult to punch many more than 600 rows in a paper tape per minute.

The connection to the electronic calculating machine is much simpler for punched tapes than for punched cards. This is related to the above-mentioned fact that the elementary read and write process for punched tapes involves only five hole positions, while upon processing a card up to 80 digits have to be handled at the same time. This also offers another advantage to the punched tapes, that the machine may process the rows of holes one by one, as soon as they have been read. This logical flexibility yields certain extraordinary advantages to the coding of problems for the machines, which will be treated in the 6. chapter of this report.

As a last advantage of punched tape it should be mentioned that the commercially available printing mechanisms, the teleprinters, are capable of producing much more beautiful printed pages than the corresponding punched card machines, the tabulators. To this it should be added, however, that also for use with punched cards there exist special printers, that are capable of producing very complicated typography (it may be mentioned that the important navigation publication The Air Almanac is now being printed by direct copying from printed pages produced by such a machine). These circumstances are not unimportant. Indeed, it is to be expected that a fast computing machine will produce large quantities of numbers that deserve to be printed. It would impose an extraordinary additional work load if this quantity always would have to be rewritten, or reset, by hand. The natural solution is that the machine itself delivers printed sheets that are suitable for purely mechanical reproduction.

Magnetic tape and wire

These two very similar media differ from punched tape and punched cards particularly in that the symbols do not produce a permanent change in the medium: a symbol may be erased and rewritten. A quantitative difference from the punched hole media is that the read and write speeds are much greater. In the machine Univac a tape system having seven parallel channels on the tape is used. Both writing and reading may be done at a speed of 10 000 symbols per second, which is about 50 times the reading speed and 1000 times the writing speed of punched tape. These great speeds, together with the possibility of overwriting of symbols, imply that these media in many cases may be considered to be auxiliary stores as much as

[1] Besk also offers reading at 400 rows per second.

media for input and output. In many machines they are in fact used exclusively in this way, since there is not other way to enter symbols on the tape or wire than through having the machine transfer them from its store. Under these circumstances the systems should be discussed in connection with other storage systems. For this reason these media will be regarded here as media for input and output only provided it is possible to transfer symbols with them through auxiliary equipment, independent of the machine.

The best example of this kind is the machine Univac, which, apart from push buttons, can only receive information from tape. Consequently the machine comes with a so-called Unityper, a machine which transfers manual impressions to tape, and a card-to-tape-converter, which automatically transfers information from punched cards to tape. A similar system is used with Harvard Mark IV, but only a machine for manual writing on tape has been built. This writer is provided with several other mechanisms that may facilitate the coding for the machine, which shall be described in chapter 6.

Undoubtedly these systems make an extraordinary efficiency of reading and writing possible. However, it should be emphasized that it will by no means always be possible to utilize this efficiency. Only with problems that involve large amounts of data will it come into its own. Thus it is significant that Univac was built to satisfy the requirements posed by The US Bureau of Census. With these applications the machine will for months on end be applied to one single problem, for example a sorting, and most of the time it will be engaged on reading and writing. With other kind of applications, for example scientific problems, most of the time will be spent on other processes, for example testing of new programs and arithmetic calculations, and it seems doubtful whether a system based on input and output on magnetic tape, with the necessity of complicated auxiliary equipment, will be a reasonable investment.

Tape and wire systems, used as auxiliary stores, will be discussed together with other storage systems in chapter 5.

Mechanisms for pure input

Every machine has to be equipped with pushbuttons for manual control of its work, a start button, a stop button, etc. What more may be transferred to the machine by direct manual control differs greatly. In this connection it is particularly important to see how the machine is started up initially. Indeed, upon termination of a problem one will usually use a ‘clear store’-button, which erases everything that was in the store. However, since the machine normally will only work under control of orders held in the store, it has to be possible to enter orders in the store by another channel than by the normal use of a read order. These orders of course will have the purpose of making the machine able to read additional orders from the ordinary read channel. But this requirement leaves wide space open for variations. Some examples may be mentioned:

Univac: A manual control causes a ‘block’, *i.e.* 120 orders, to be transferred from a previously selected tape to the store, upon which the machine immediately starts to execute the first one of these 120 orders.

Edsac: A manual control inserts a fixed set of 42 orders from an electromechanical selector in the first positions of the store, whereupon the machine immediately starts to execute the routine specified by these orders. This routine takes input of orders that are punched in the tape according to a carefully planned, very flexible code, and puts them in the store. This code will be described in detail in chapter 6.

Besk: Pushbuttons make it possible to insert an arbitrary word in an arbitrary position in the machine, and to instruct the machine to execute the order at an arbitrary address. By insertion of a read order the machine is made to read a word from the tape. This may contain two read orders, and thus the process may develop automatically.

A discussion of the possibilities indicated here is best postponed until chapter 6.

Among further possibilities of manual controls it may be mentioned that telephone dials in several machines is used as input organ.

Mechanisms for pure output

With few exceptions the common practice has been to equip the machines with possibility for giving output directly through an electric typewriter or teletyper. Apparently it has appeared that there always must be a means of having a single result printed at once, without intermediate link. In this connection the experience from the Edsac is instructive. This machine originally was provided with a teletyper as its sole means of output. Later it was decided that the machine also had to have a tape punch. After a short period with experience of having both systems available it was decided to disconnect the directly controlled teletyper. Instead the tape is led from the punch directly to the teletyper, and so the results will be typed after only a short delay. This system is satisfactory to everybody, and no lack of a directly coupled typewriter is felt. The positive advantages of the system are the greater speed of the punch and the possibilities of feeding results back into the machine and of obtaining several copies of the printed results.

As far as its has been reported, all machines are equipped with cathode ray tubes that at any time will allow direct reading of the machine contents of numbers and orders. Although they are not normally designated as such, these tubes may of course be considered to be output mechanisms. This principle has been applied more extensively in Whirlwind I. This machine is provided with cathode ray tubes that may be used to present curves of data held in the machine. Merely by taking a photograph of such a picture one may often obtain sufficient information about a set of results that otherwise would require printing of a long table.

In this context a few remarks about the results produced by a machine, as viewed more generally, may be in order. Considering the treatment of materials of numbers that one wishes to perform, one may perceive a general tendency, namely that vast bodies of data are reduced to a few numbers, sometimes just one. This process is characteristic of the scientific method, where the vast body of single results is seen as the expression of a single law of nature with a few parameters. It may however be noted that the same tendency may be found in other areas, where no scientific justification of the process may be found. Perhaps one may relate the tendency to the restricted perception of man. It appears that the only thing a person is able to perceive directly is a number having one, or at most two, digits. Whenever numbers having more digits appear they have to be compared with other numbers of the same order of magnitude. Here again the perception appears to rest on the difference having only a few digits. If this is correct it is a circumstance that should be taken into account in the work with computers. Indeed, the goal should be that the computer delivers a result that may be perceived directly by a person. In this case the machine in principle need only communicate a few two-digit numbers. In scientific problems this might *e.g.* be the mean deviation between observations and theory.

From this point of view all extensive bodies of numbers delivered by the machine, such as tables, are merely incomplete results. Only if the numbers are processed or used further a perceivable results will be arrived at, and it then appears to be unreasonable that the machine did not already perform this further application.

Storage systems and internal function

The decisive aspect of the internal function of the machines is the speed (access time) of the working store. The machines under review are provided with three types of working stores: 1) Williams tubes, access time around 10 microseconds, 2) Mercury delay tubes, access time around 500 microseconds, 3) Magnetic drums, access time around 10 000 microseconds. In addition to the working store many machines are equipped with one or more auxiliary stores in the form of magnetic drums or tape systems. Examples of these combinations are:

Fast store (either 1) or 2)) and magnetic drum:

The machine of the Institute of Advanced Study
Swac
The Ferranti Computer
IBM 701 (also magnetic tape)

Fast store and magnetic tape or wire:

Edsac
Seac
Whirlwind I

Cadac 102-A has working store on magnetic drum and auxiliary store on magnetic tape.

The issues of the internal functions of the machines will be discussed in the following order:

1) Parallel versus serial processing of numbers.
2) Number systems.
3) Word length.
4) Elementary orders.
5) The internal code.
6) Auxiliary stores.

Processing of numbers

The advantages of number processing that treats the digits in parallel at the same time are:

a) The possibility of high speed of arithmetic operations.

b) The machine will be composed of a large number of like components, one for each digit position. Thereby the maintenance and the detection of faults is facilitated.

Clearly the high speed can only be exploited fully provided the store has a correspondingly short access time. Parallel processing therefore is a good match to a Williams tube store, which also for other reasons is best suited for parallel processing.

The advantages of serial processing are:

a) The modest amount of hardware required in the arithmetic unit. An adder for serial numbers is well known to be a fairly simple unit.

b) The simplicity of a register for holding a single serial number. While a register for parallel representation requires at least a couple of tubes per digit, a recirculation register for a serial number may be constructed with a few tubes together with a suitable delay line.

These circumstances are decisive in the selection of the number processing. Indeed, if a fast working store is available (a) it is desirable that arithmetic operations and number transfers are performed as fast as possible, and (b) there is not need for a large number of auxiliary registers.

If on the other hand the working store is relatively slow (a) it is unreasonable to make a great effort to speed up the individual arithmetic processes, and (b) there is great interest in having many fast registers available.

This last issue turns out to be highly important. This is related to practical experience concerning the operations that are used in an electronic computer. It turns out that mostly the machine is engaged on only a few orders and numbers at a time. Suppose for example that the machine is engaged on computing a square root with the aid of an iteration cycle. It will then execute a short loop of orders many times, and operate only upon a few different numbers. When the iteration has converged it will usually be necessary to execute various control operations, for example such that make the machine return to the main routine and make ready for starting the work in another subroutine. However, the essential point is that these operations, which of course may involve any locations of the store, only are executed once when the orders of the iteration loop have been executed perhaps five or ten times. It is therefore clear that provided there is a sufficient number of fast registers available to accommodate a complete loop and the numbers that it works on, then such a loop may be executed completely at a speed which is independent of the store.

Considerations of this of this kind are now widely appreciated by constructors of smaller, cheaper machines, which have their working store on magnetic drum. As illustration some details from three such machines will be given:

Cadac 102-A. The machine is provided with a magnetic drum holding 1024 words of 42 binary places, with an average access time of 12.5 milliseconds. In addition it has 8 registers with an access time of 1.6 milliseconds, the so-called buffer registers. These 8 registers are also held on the drum, but *two* magnetic heads are used, one which always reads and another one, positioned one eights of the circumference before the first one, but in the same track, which always writes. The pulses are always sent from the first head to the second one. The two heads and the drum thus together form a delay line. These 8 registers may either be used just as the other store locations. They may hold numbers or orders, in the usual way. But they may also be filled all at once, by a single order, from an arbitrary place of the main store. The machine works with a three-address code and the intent is that normally both the two operands and the result will be placed in the buffer registers. If so most operations can be executed in less time than a half revolution of the drum, and it is possible to place adjacent addresses diametrically opposite on the drum. If so a complete tree-address operation is executed in 12.5 milliseconds. If no buffer registers had been available the execution time would be tree times as long.

Cec 30-201 has a main store of 4000 words of 10 decimals, access time 8.5 milliseconds. In addition it has four special groups of each 20 locations, arranged in four delay lines with access time of 0.85 milliseconds. The user has a choice of two modes of operation of the machine: A) The order to be executed is always taken from a definite one of the four special groups of 20 locations. Within this group the orders are taken in normal sequence until a transfer of control is made to an address in the main store. When this happens the order to which transfer of control is made, together with the following 19 orders in the main store, are automatically copied to the 20 locations of the special group. The remaining three special groups may be used in the ordinary way as fast access registers. B) With this mode of operation no special locations of control are distinguished, and the order to be executed may be taken from any place in the store. Block transfers are made only if explicitly ordered.

Elliott 401, Mark I. The store uses a magnetic disk, having 23 tracks of 128 words of 32 binary positions. But the addresses run only from 0 to 1023, corresponding to 8 tracks. So as to utilize all the tracks the last 128 addresses may be coupled to any of 16 different tracks. Thus the addresses 0 to 895 permanently identify definite places on the disk, while the remaining 128 addresses, depending on a previous execution of an order that selects one out of 16 magnetic heads, each identifies 16 different places on the disk. This selection of the last 16 magnetic heads takes place by means of mechanical relays, and the execution of a selection order must therefore be followed by a suitable delay time before the addresses 896 to 1023 are again referred to. In addition to the store there are the following registers that are held in magnetostrictive delay lines: accumulator, multiplicand (may also be used as a general fast register), two registers X and Y, both fast and for general use. The control system in this machine is rather unusual and allows both unusual versatility of each operation and high speed, although at the expense of the convenience of coding. Thus if the operands are placed in the most favorable positions it is possible to execute an addition every 200 microseconds.

The examples given here do not exhaust the possibilities. However, it appears that it must be possible rationally to arrive at the most advantageous system of this kind. The decisive factors will be partly the cost of each extra register that is put into the machine, partly the kind of operations that one wishes to be able to perform at best advantage. It should be possible to analyze the various possibilities simply by investigations of the way concrete tasks, particularly subroutines, would be solved under the various conditions. There is hardly any doubt that a well designed machine of this kind, with a main store on magnetic drum, in speed will be competitive with machines provided with mercury delay lines, but without auxiliary registers.

Number systems

In the internal operations the machines described here present only two different systems, decimal and binary, the last with the derived octal and hexadecimal. The arguments for the decimal system are a) that no conversion is needed when ordinary numbers have to be processed by the machine, and b) that it is easier to follow what happens in the machine. In particular it supposedly should be easier to trace errors.

The arguments for the binary system are: a) It is the most economical system. With this system the logical combinations of the yes-no elements are exploited fully, while with the decimal system they are only partially used. It is well known how even in a machine that works decimally the information is represented by yes-no possibilities (coded decimal system). For example a ten-digit coded decimal number is represented by 40 yes-no elements. However, a number of 40 binary digits yields an accuracy better than 12 decimals.

A discussion of the two possibilities properly has to be seen on a background of a more comprehensive discussion of two basic views, that in pure form take the following forms: A) The machines ought to be provided with many special automatic functions so as to facilitate the work of the users. B) The machines should be equipped with a few simple basic functions, more complicated functions are built from the simple one by means of subroutines. In favor of A speaks that automatic equipment will operate faster than subroutines. On the other hand, subroutines require no maintenance and may perform indefinitely complicated tasks. The two directions each have their proponents and in the two camps there is clearly a difference also with respect to the character of the problems. In problems where the main difficulty lies in the amount of data to be processed, *e.g.* statistical and commercial problems, while the operations to be performed are relatively stereotyped, one will tend to prefer more complicated equipment, which has the particular properties that are needed built into it. On the other hand, if one is interested in scientific problems over a wide range, including *e.g.* logical and methodical experiments, one will prefer a more general machine. In that case a comparatively simple machine will be most useful.

As the first example of this kind we may compare the decimal and binary systems. When building a machine with special regard to business problems there is good reason to choose the decimal system, since the work will involve large bodies of data, given in the decimal system. But if one is interested in calculations in group theory it is likely that one will deal with data that at no stage will be understood as numbers. Even if such a calculation undoubtedly might be carried out in any machine, irrespective of its number system, it is likely that the use of the decimal system would entail some loss of the capacity of the machine, without any advantage.

Word length

The information in a word held in the store of the machine will have two different meanings, depending on whether it is taken to be a number or an order. Consequently two different requirements are imposed on the length of the word. On the one side one usually wishes to be able to work with a number precision of about ten decimal digits (corresponding to about 35 binary digits). On the other side the word length has to match the length of the elementary order. The elementary orders are described below. It is found that these conveniently may be held in about 18 binary digits in case of a one-address code. Thus it is clear that in a machine which works with a one-address code it is convenient to put two orders in one word, or, to the same effect, to let the order fill a word, but offer the option to perform arithmetic operations upon double words. This system is used in most machines that work with a one-address code, including Univac, Edsac, and Besk. An exception is Whirlwind I, which only works upon words of 17 binary digits. In machines that work with a three or four address code an order may conveniently be put in one word.

Elementary orders

The elementary order that are built into the computers may be divided into the following categories: arithmetic (including round-off), shift (left and right), move (from store to registers, etc.), modification (logic product and the like), control jump (conditional and unconditional), input, and output. Within each category it is possible to specify a certain logical minimum of capability which must be provided. For example, it is possible to perform all arithmetic processes merely with the aid of a subtraction order together with shifts and moves. However, one will always prefer to automize far more processes than logically necessary. In this way time is gained at the cost of additional complication of the machine. This adding of complications has to stop short at some point, which mostly is selected according to the wishes of the constructors. As elaboration of this issue the various categories of orders shall be discussed, with special attention to the possible variations.

Arithmetic orders. All of the machines described in this report have orders that perform additions, subtractions, and multiplications. Most of them also have a division order. The variations among the machines relate to the following possibilities: A) Operations upon numeric values: may/may not be done automatically. B) The multiplicator is unchanged/has disappeared, after a multiplication. C) Product sums: may be formed directly/the accumulator is cleared before multiplications. D) There is a special order for multiplication with round-off/round-off requires an extra order. E) A possible clearing of the accumulator takes place: at the beginning/at the end of an order cycle (see also move orders).—Three address machines operate always directly with store locations and so for them all remarks about registers are irrelevant.

Shift orders. The machines are normally provided with elementary orders for arbitrary right and left shifts of the number in the accumulator. Variations: a) There is/is not the option to shift the digits of the number independently of the sign. b) There is/is not an order for left shift of the number to the most significant position.

Move orders. The machines are normally able to move numbers in arbitrary directions to and from storage locations and registers. The special possibilities relate to what registers are present in each machine. The most important possibility of variation relates to the time at which the registers involved are cleared, which may be placed at the beginning or the end of an order cycle. In three-address machines move orders are irrelevant.

Modification orders. Most machines have an order of the type: find the logical product of two numbers (*i.e.*, in binary machines, write 1 in the digit positions where both numbers have 1 and 0 in all other digits positions).

Control jumps. Most machines may execute unconditional jumps and two complementary kinds of conditioned jumps. It is further common that the register, whose value conditions the jump, may be cleared or left uncleared in consequence of the jump. Finally there are usually also control jumps that are conditioned by *overflow* in the accumulator, *i.e.* that the capacity of the accumulator has been exceeded by continued additions. In three-address machines the conditioned control jumps usually depend on a comparison of two numbers in the store.

Input. The function of the input orders essentially depends upon the input mechanisms that are available. Obviously there must be at least one order for each mechanism. Only a single issue of general import shall be mentioned: There will always be one input channel which is distinguished by being normally used for input of programs. It follows that the notation, the external code, which is used when a program is recorded outside the machine, in a decisive manner will be dependent upon the input order of this special channel. With punched tape, a commonly used order input medium, the choice is between an order which reads a single row of holes and one which reads a complete word, or even several words. In chapter 6 this question will be discussed further.

Output. The output order, like the input orders, are essentially dependent upon the mechanisms that are attached to the machine.

The internal code

The internal code shows how a word held in the store of the machine will be interpreted by the machine as an order. In chapter 6 it will be described how this code relates to the external code, that which is used in writing during the programming. The properties of the internal code is displayed in the way the digits of a word influence the function of the order.

One-address code. With this type of code each elementary order specifies an operation that will at most involve one storage location. Each order of the internal code must accommodate one address and a specification of the operation in question. With a storage capacity of 1024 words the address requires 10 binary digits. The operation part usually requires at least six binary digits. These are sometimes conveniently divided into two groups: a main part and one or more modification digits. Examples:

Edsac. An order has 17 binary digits, of which 16 are used in the following way:

```
Binary digit no. 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17
                 «............»    «.......................................»
              Main operation              Address
```

The address includes the binary digits 7 - 16, and the operation part the digits 1 - 5 and 17. No. 6 is not used. No. 17 is a modification digit. It is used to indicate whether the operation specified by no.s 1 - 5 is to be done upon a short or a long word. The main operation part gives the possibility of 32 orders. Of these only 18 were used until the installation of the system for magnetic tapes: 1) addition. 2) Subtraction. 3) Store to multiplicand register. 4) Multiply positive. 5) Multiply negative. 6) Accumulator to store, clear accumulator. 7) Acc. to store, no clear. 8) Logic product. 9) Right shift. 10) Left shift. 11) Conditional control jump a. 12) Conditional control jump b. 13) Unconditional control jump. 14) Input of row from punched tape. 15) Output of a character. 16) Round-off. 17) Stop. 18) Mix order (special addition, useful for modification of orders).

Besk. An order comprises 20 binary digits:

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
   «.........................................»    «...»  «.................»
              Address               Modification  Main operation
```

The binary digits 1 and 13 are not used. The modification digit, no. 14, specifies clearing of the accumulator *before* the operation, , no. 15 specifies long or short word in the operation. Of the 32 main operation possibilities, the following 20 were used in October 1953: 1) Addition. 2) Subtraction. 3) Addition of numerical value. 4) Subtraction of numerical value. 5) Accumulator to store. 6) Sum of store and accumulator to multiplicand register. 7) Difference of store and accumulator to register. 8) Multiplication, full length. 9) Rounded multiplication. 10) Multiplicand to accumulator. 11) Left shift to most significant position. 12) Left shift. 13) Right shift. 14) Logic product. 15) Conditional control jump a. 16) Conditional control jump b. 17) Control jump upon overflow. 18) Input of word from tape.19) Output of decimal digit on typewriter. 20) Output of symbol on typewriter.

The Ferranti Computer. This machine deserves special mention since its internal code provides unusual possibilities. These possibilities derive from the addition, immediately before the execution of an order, of the value of one out of 8 special registers to its value. These registers in the machine are called ‘the B-line’. There are three binary digits in an order, in addition to the usual address and operation digits, that specify which of the 8 B-line registers should be added to the order before it is executed. The advantage of the system is that it makes a large part of the usual address modifications superfluous.

Three-address code. Here the digits of the orders may be divided into three addresses and a function part. This then requires about 3 * 10 + 6 binary digits = about 36 binary digits in the order. There is no question of the elementary orders referring to the registers of the machine, since the order accommodates the specification of two operands and one function.

The issues in comparing one-address and three-address codes are:

The advantages of the one-address code: Logical economy. This manifests itself in both the construction and the use of the machine. The control system undoubtedly may be built simpler with a one-address code than a three-address code. In use there will probably be needed less information for specifying an operation with a one-address code, since several types of operation only need a single address. One-address code is also economic in that the arithmetic registers of the machine always are available as storage locations.

The advantages of the three-address code: a) A word length of approx. 10 decimals is suitable for holding one three-address order. b) The time needed in the machine for fetching the orders from the store is less, since more operations are executed by each order.

For a user of the machine there is hardly any decisive difference between the two systems. Thus it may be concluded that a one-address code is to be used when the access time is relatively short, and a three-address code when the access time is relatively long, compared to the time of arithmetic operations.

Auxiliary stores

The working store in machines equipped with Williams tubes or mercury delay lines will usually be limited to a few thousand words. In such machines it is common and advantageous practice to extend the internal store with an auxiliary store. The order list of the machine will then include orders that transfer a sizable quantity of data from the auxiliary store to the working store or vice versa. Thus before being used the information held in the auxiliary store has to be transferred to the working store.

In comparing an auxiliary store held on magnetic drum to one held on magnetic tape the following issues are relevant:

Advantages of magnetic drum: 1) A drum system possibly is easier to construct mechanically than a tape system since it involves no accelerations. 2) The access time is comparatively low. 3) It may hold permanent subroutines.

Advantages of magnetic tape: 1) The system has unlimited capacity. 2) Each user may have his own tapes. This is important because the use of a drum forces the user to fill it each time it is used.

As a special example it may be mentioned that the designers of the Edsac, after having first completed the development a drum decided not to use it at all, and instead constructed two tape units for the machine. One of these permanently carries a tape holding selected subroutines, while the tape on the other unit may be changed.

Organization of program preparation and error diagnosis

In this chapter certain problems that arise in the practical use of electronic computers will be discussed. In addition a detailed description of the coding techniques used with the Edsac will be given.

In using an electronic computer for the solution of a computing problem the following steps may normally be distinguished. The problem is presented in abstract mathematical form. Through numerical analysis it is converted into a computing problem, involving a finite number of computing processes. This computing problem is coded in the code of the available computer, *i.e.* it is divided into a series of computing operations of the kind allowed in the external code of the machine (*i.e.* the code which is used in writing a program for the machine). The program formed in this way is then transcribed manually into one of the input media of the machine (it is punched in tape, or the like). Finally this medium is placed in the corresponding input mechanism of the machine, and the machine is instructed manually to read the program inscribed in the medium and to perform the corresponding processes. During this read process the program held in the medium in external code is converted to the internal code of the machine.

The use of an electronic computer does not present any difficulties of principle. However, experience shows that great difficulties arise from the need to perform the operations that enter into the programming completely correctly. Even experienced users of computers have to exert extreme care in order to succeed in generating a correct program in the first attempt. It further appears that the errors that are committed often are of quite elementary nature. For example, at a certain stage of the programming it has appeared that erroneously an order has been omitted from the program. This order then has to be inserted at its proper place. It is then necessary to move all following orders from their present location to a location address one higher. In addition one has to change the address in all orders that refer to one of the orders that is being moved. It turns out that this kind of changes often give rise to errors.

Because of this practical difficulty the planning of the coding work must be considered to be a matter of primary importance. As a matter of fact, the efficiency by which a user may exploit the machine depends to a high degree on the organization of the programming. The important factors here are: a) the external code being used, b) the organization of a library of subroutines, and c) the set-up of the methods for diagnosing errors.

a) The starting point of an analysis of the coding system is the existence of two codes, the external one and the internal one, which were mentioned above. These may be described as follows: The internal code is the one held in the store of the machine during the solution of the problem. Properly speaking this may be written down on paper after having been recoded to an external code. Indeed, in its initial form it is held in the form of electrical charges or pulses, or magnetic fields, and in order to write it down there will always be required a set of conventions that bridge the transition from the state of the electrical system to symbols that may be written down, that is to an external code. This consideration shows immediately that the external code rests on pure convention, and that therefore one has the option to choose the external code in such a way that the coding work is facilitated as far as possible.

As further explanation of this situation we shall now give an account of the coding systems for two machines, Besk and the Edsac.

Besk. The word length in this machine is 40 binary digits, which gives room for two one-address orders. The structure of each of these two orders has been explained on page 15. Besk is equipped with a mechanism for input of symbols from 5-hole punched tape. The elementary input order has the effect that one word is read from the tape into storage location *n.* It must be noted, however, that only four of the five hole positions of the tape are used. Thus one word corresponds to 10 positions on the tape. In the coding one has desired to maintain a close connection between the internal and the external code. Therefore one has had to work with a sexadecimal number system, corresponding to the four hole positions on the tape. Take as example the following instruction pair:

```
0000’0001’1100’0111’0000:0000’1110’0100’0010’1011
↑—————————↑  ↑↑↑ — ↑   ↑————————— ↑   ↑↑↑——— ↑
 Address = 28     ↑↑   ↑     Address = 228    ↑↑   ↑
      Clear acc.→ ↑↑   ↑   Do not clear acc.→ ↑↑   ↑
     Whole word→   ↑   ↑         Whole word→   ↑   ↑
                  Addition                   Subtraction
```

These two instructions will have the effect that the whole word at location 28 will be transferred to the accumulator and that the whole word in 228 then will be subtracted.

In order to use the elementary input order for reading these instructions they first have to be divided into sexadecimal parts, as indicated by the apostrophes. Then each part is written sexadecimally. For this purpose the following symbols are used:

decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
sexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F.

We then get for the two orders above:

01C70:0E42B

This has to be punched into the tape and will then be ready for input.

The system just described is the one used for coding of Besk. The system retains a very close correspondence between that which is written on paper and that which may be read directly as charges in the store of the machine. This close correspondence is considered by the programmers of Besk to be a decisive advantage.

Edsac coding. The word length is 17 binary digits. However, two consecutive words and a single store pulse between them are joined together to form a long word of 35 binary digits. But since long words are used only for numerical quantities this possibility may be ignored when describing the coding. The structure of an Edsac order is shown on page 15. The input mechanism of the Edsac is, similarly as with Besk, a punched tape mechanism for 5-hole teletype tape. However, the elementary input order differs from the one available in Besk by providing for the reading of only row of holes at a time. Further all five hole positions on the tape are read, not only four as in Besk. The corresponding five-hole symbol is place by the input order in the five rightmost positions of a storage location.

It is clear that this input order is capable of transferring instructions that are punched in the tape according to the external code, to corresponding locations of the store, only if it is combined with other orders. Thus the input of orders requires the action of a complete program. This might at first sight appear to be an undesirable complication. In reality this offers a flexibility and convenience of the external code which is a great advantage to the programming.

For the Edsac two different order input programs have been used. The first one was used only for a short period when the machine had been completed, and was quickly replaced by another one, which offers far more advantages to the users of the machine. It is instructive to describe the mode of operation of both programs.

The first input program for the Edsac read an external program in which each order had three parts: First the function letter (one letter out of 32, written as the usual letters, supplemented by a few Greek letters), then the address, written decimally, without superfluous zeros, and finally a terminating letter, which might be either S or L, according to whether the operation had to work on a word of 17 binary digits (short) or of 35 binary digits (long). As illustration we may use the two orders above. In the internal Edsac code they will be written

```
11100’00000011100’1
↑— ↑ ↑ ————————↑ ↑
  Add     Address 28    Long number
01100’00001110010’1
↑— ↑ ↑ ————————↑ ↑
Subtract  Address 228   Long number
```

In the first external code they were written as follows:

```
A  28 L
S 228 L.
```

In the punched tape these symbols are punched in exactly the same way: A28LS228L.

We may now describe the operations that the input program had to perform while reading such a program:

1) Read the first symbol, A. Shift it eleven binary positions to the left so as to make it occupy the proper position of a function letter in a word, and then store it away temporarily.

2) Read the next symbols on the tape as long as these are digits from 0 to 9. These symbols should be combined so as to form the address (*i.e.* when the second digits arrives the first one must be multiplied by 10 and added, etc.), but when one of the symbols S or L is encountered the function letter and perhaps a unit in the last position (corresponding to L) is added, to make the order ready to be placed in the store.

The decisive features of the use of this external code are as follows:

a) The function part is a quantity of 5 binary digits and may therefore be represented exactly by one row of holes on the tape.

b) The address in the external code is written and punched decimally in the tape. Thus each decimal digits exploits only 10 of the 32 possibilities of a punched row. Therefore it is possible to detect the arrival of the terminal letter merely by its value. (The symbols S and L correspond to hole rows equivalent to decimal 12 and 25). Thus it is not necessary to keep track of how many digits of the address have been read, and it is not necessary that the address is always written with the same number of digits.

The second order input program that was constructed for the Edsac takes over the essential features of the first one, but augments it with further features that both exploit the logical possibilities better and make the work with a library of subroutines more convenient.

The most important problems in working with a library of subroutines are as follows:

1) A subroutine must be ready to enter into any program and must therefore be prepared to be placed anywhere in the store of the machine. It will often be the case, however, that orders within the subroutine itself makes reference to locations within the subroutine itself (*e.g.* when a subroutine modifies its own orders). The addresses of these orders must therefore be modified according to where in the store the subroutine is placed.

2) The subroutines must be constructed to cover as varied conditions as possible. The necessary flexibility requires that it is possible for each user to specify one or more parameters in the subroutine. It is therefore important that the library system allows such parameters to be inserted in the subroutines.

In the second order input program for the Edsac a convenient solution of both of these problems is achieved through an extension of the meaning of the terminal letter. While the function letter and the address are read in the same way as in the first program, the number of different terminal letters is extended from two to fifteen. Two of these, F and D, take over the function formerly served by S and L. The remaining ones act as follows: to each letter corresponds a definite location in the store (viz. to H no. 45, to N no. 46, to M no. 47, etc.). When one of these letters is used as terminal letter in an order, the quantity held at the corresponding location in the store will be added to the remaining parts of the order immediately before the order is placed in the store. Example: Assume that the quantity

```
00000’00000000010’0
↑ — ↑ ↑ ———— ↑ ↑
  P        2      F
```

which in the new external code is written P 2 F, is placed in location 45, that location which corresponds to the terminal letter H. Then every order that in the external code has H as terminal letter will appear in the internal code with an address which is 2 greater than in the external code. Thus while

```
        P 4 F    produces    00000’00000000100’0
we get from
        P 4 H:               00000’00000000110’0
```

We see that the terminal letter provides a means of distinguishing certain orders. In particular it is convenient if an unspecified parameter is to be inserted as address in several orders in a subroutine. It is then necessary merely to distinguish the orders involved by a distinct terminal letter. If the value of the parameter is placed in the corresponding store location immediately before the subroutine is read, it will automatically be added to the proper orders in the subroutine. In this way it is possible to leave twelve different parameters open in a subroutine.

A minor detail deserves discussion: The terminal letters in the external code of the Edsac have the consequence that there is no one-to-one transition from the internal code to the external one. In terms of the example above it is seen that P 6 F and P 4 H will produce identical orders in the internal code. This is claimed by certain people to be an essential disadvantage. It has in fact the consequence that a check by comparison of the external code as written and that which is held in the store of the machine cannot be made directly. However, as discussed further below, the particular check procedures that are used with the Edsac make an advantage, rather than a disadvantage, of this feature of the system.

As illustration of the use of the terminal letters in the subroutines of the Edsac library we shall give two examples: Subroutine T3 has the title: General cosine. In the specification sheet for T3 we get the following information: The subroutine computes (1/2)cos $x \infty 2m,$ where x is the number found at location 4 when the subroutine is called. The parameter m is chosen by the user. It should, when the subroutine is read into the Edsac, be found at location 45 in the form of the order P $2m$-3 F.In this way this parameter will automatically be inserted in the subroutine at the place, or the places, where it is needed, by virtue of the orders in question having been provided with the terminal letter H (corresponding to location 45). Thus if we need the function (1/2)cos 32 x, $m = 5$. If we further assume that the orders of the subroutine should be places at the locations 150, 151, etc. we have ourselves to punch the following symbols in the order tape:

T	150	K	Standard symbols for placing the orders
G		K	from location 150 onwards.
T	45	K ⎫	Place the parameter $2m$ -3 = 4 at location 45.
P	4	F ⎭	

The orders of the subroutine must be punched in immediate sequence hereof. This is done by a simple automatic copying of the tape of the subroutine, which is kept in the library.

Subroutine G1. Title: Integration of simultaneous first-order differential equations by modified Runge-Kutta process. This subroutine performs all the arithmetic processes that are needed in the stepwise integration of an arbitrary number of simultaneous differential equations. The subroutine itself contains no information about the form of the differential equations. These have to be specified for the machine in the form of an auxiliary subroutine that the user has to provide. Let us denote the variables $y_1, y_2, \dots y_n$.. These have to be placed, when an integration step is initiated, at a sequence of locations at addresses a-n+1, a-n+2, …a-1, a. The auxiliary subroutine required is a closed subroutine that computes the derivatives, y_1', … y_n', when $y_1, \dots y_n$. are placed in these locations. In other words, it has to compute the functions

$y_1' = f_1(y_1, \dots y_n.)$

…

$y_n' = f_n(y_1, \dots y_n.)$,

and then has to place the quantities $2^{mh}\, y_1'$, … $2^{mh}\, y_n'$, where m is a parameter, in the n locations b-n+1, b-n+2, …, b. The auxiliary subroutine, which has the sole task of computing the functions f_1, f_2, etc., is placed with its first order in location d. When this subroutine is available G1 will work in the following manner: each time it is called all the variables y_1, … y_n. will be advanced one step, corresponding to an increase of the independent variable of h. During this work the subroutine also needs n auxiliary locations, c-n+1, c-n+2, …, c. In G1 6 different terminal letters are used to specify the parameters. These have to be provided by the user in the following form:

Terminal letter	Punched by the user			Meaning
	T	p	K	The 68 orders of G1 should be placed
	G		K	at locations p, p+1, …
	T	45	K	Standard control symbol
H	P	a	F	Address of y_n..
N	P	n	F	Number of variables.
M	P	$b - a$	F	b is the address of y_n'.
Δ	P	$c - b$	F	c is the address of the last aux. location
L	P	2^{m-2}	F	Normalization factor.
X	P	d	F	The auxiliary subroutine starts in d.

Immediately following these symbols the orders of G1 have to be placed, through an automatic copying.

The coding system which has been described has been developed from the following assumptions:

1) The varied and effective use of a computer depends on the existence of a comprehensive library of subroutines.

2) If the user of a subroutine shall have complete freedom in the choice and combination of the subroutines he needs for his problem, and if the subroutines shall provide for parameter values given by the user, there is an strong need for a system that will provide for automatic additive modifications of orders.

The latter conclusion is challenged from various sides. It is claimed that in a machine equipped with an auxiliary store in the form of a magnetic drum or a magnetic tape system, the difficulties are solved if all subroutines are permanently held on the drum or the tape. In such a system they would never be held in the working store of the machine over a long time, not even during the solution of problems in which they are used repeatedly, but would always be called in from the drum or the tape immediately before they were needed.

Undoubtedly such a system would be very convenient in use. However, the question is whether it would not generally put a heavier load on the capacity of the auxiliary store than desirable. It may be mentioned that the library of the Edsac currently holds about 120 subroutines with a total of about 6000 orders. Several of these subroutines will be used so rarely that they need not be held in the auxiliary store. As soon as the need for holding certain subroutines outside the machine is admitted, a code system of the Edsac type becomes again highly desirable.

A standard system for additive modifications of orders during the input process is, as shown above, practically an indispensable necessity when the work with the machine has to be based on a library of subroutines. As will now be explained, further convenience is achieved when coding within such a system. This will be explained most easily in terms of an example that accounts for the processes that enter into the coding of a complete problem in the Edsac system. Let this problem be the following: Two simultaneous differential equations are given:

$$y_1' = y_1 \sin y_2$$
$$y_2' = e^{-y_1}$$

What is wanted is a table of the solution of the equations that passes through the point $(x, y_1, y_2) = (0, 0.75, 0.84)$, with steps of 0.01 in x.

The most important processes in the solution of this problem are the following:

a) Establishment of a rough survey of the necessary subroutines and numbers. The following subroutines are needed: 1) A number input routine, *i.e.* a subroutine that will perform the conversion of the numerical initial data, that is the numbers 0.75, 0.87, and 0.01, from decimal form into binary. 2) An integration subroutine, for example G1 described above. This requires 3) an auxiliary subroutine that computes y_1' andy_2' when$y_1$and y_2 are given. The auxiliary subroutine will need 4) a subroutine for computing sin x and 5) a subroutine for computing e^x. Finally we need 6) an output routine that will convert the results to decimal form and will perform output into a suitable medium. 7) The work of these subroutines has to be controlled by a main routine that performs activation of the various subroutines in the right order and that places the necessary numbers in the locations required by the work of the subroutines. The numbers that enter into the problem are merely those that are required in the work of the differential equation subroutine, *i.e.* y_1, y_2, y_1', y_2', plus two auxiliary quantities. If one wants to have the independent variable, x, printed one has to include a location for its current value and one for the step length.

b) Each of the independent subroutines must have a terminal letter assigned to it. In doing this assignment one has to keep an eye on the terminal letters that are used locally within the individual subroutines, since each terminal letter may only serve one function at a time. With the twelve terminal letters that are available this does not cause any difficulty. We may, for example, assign the terminal letters in the following way:

Input routine (approx. 40 orders)	G
Integration subroutine (68 orders)	A
Auxiliary routine (20 orders)	X
Sine routine (60 orders)	B
Exponential function (50 orders)	C
Output routine (50 orders)	V

The placement of the numbers is required by the integration subroutine G2 to be fixed with the aid of the terminal letters H, M, and Δ. This assignment of terminal letters to the different routines is done by placing the address of the first order of each subroutine in that location of the store that corresponds to the terminal letter of the routine. After this has been done the address of the first order in the input routine may be referred to as 0G, or just G. The second order in this subroutine will have the address 1G, etc.

It will be seen that in virtue of this use of the terminal letters one may refer to the various parts of the program before being committed to the specific locations that the routines eventually will occupy in the store. The definitive placing of the subroutines may be postponed until the last minute. In a relatively simple problem as the present one this is only a minor matter, since there is plenty of room in the store. However, in problems that are so complex that it is necessary to utilize the store capacity fully, it is highly important. The point is that in working out the various parts of the program one has to be thrifty of storage locations. In practice the development of such a program will inevitably take the form of a succession of improvements and simplifications of the program. Each time one has in this way achieved a saving of a storage location, all the following orders in the store have to be moved one location ahead in the store. All orders whose address refers to these orders must therefore have their address part modified. When using the terminal letters it is achieved that a complete subroutine may be moved in the store merely with the change of a single parameter. All other address modifications will automatically be performed upon input of the program into the machine.

c) For the parts of the program, either subroutines or main routine, that cannot be taken from the library, one now has to write the elementary orders. In addressing the orders within each of these parts one may always start by assigning the first order the address 0. By using the terminal letter of the routine it is in fact possible to refer to orders whose addresses are known only relatively to the first order of the routine. Thus it is possible to write the orders of the routines completely, without having settled their absolute placements in the store.

For each of the library subroutines one has to write the corresponding parameters, in accordance with the instruction belonging to each subroutine. When all parts of the program have been written it is possible to determine how many orders are necessary. Only at this stage it is necessary to fix the absolute placement of the parts. This placement is specified merely by the addresses that are put into the locations that correspond to the terminal letters. For example, the sine subroutine will have its first order placed in location 210 if the quantity P 210 F is put into location 53, that location which corresponds to the terminal letter B. Since the sine routine has 36 orders, the next subroutine has to be placed with its first order at a location whose address is 246 or higher.

d) It will often be convenient already during the first planning to think of including a checking routine in the program. As already stressed, it gives often great difficulty to eliminate elementary errors in programs. The library therefore provides a series of routines that merely serve to help the users detect errors. They are used only as long as the program is incorrect. As soon as all errors have been removed the checking routine is also taken out. There are many kinds of checking routines to choose from. They are all alike in that they let the programs of the program be executed one after another, in exactly the same way as they would if the checking routine were not present. However, mixed in between the execution of the orders of the program the checking routine provides for printed output of suitable check information through the output mechanism of the machine. For example one will often use checking routine which provides for output of the function letter of that order in the program which has just been executed. Clearly the sequence of letters produced by the machine in this way will be a convenient means of checking whether the machine has followed the intended sequence of operations. Other checking routines provide output of the numerical values that are processes by the program.

The use of checking routines has proved to be an effective and quick way to localize errors in a program. One avoids the need for users to let the programs of the machine be executed slowly, one by one, so as to make it possible to determine where deviations from the planned route take place. The checking routines will of course slow down the program, but it turns out that this will be executed at about five orders per second. Clearly it would be impossible to follow the work of the program by watching the cathode ray tubes at this speed.

e) The written symbols, both the orders in the newly constructed subroutines and the specifications that are necessary in using the library routines now have to be punched in tape according to the Edsac code. This may be done directly, using the punching machines that are available. As protection against errors of this transcription one normally performs this punching twice. The two punched tapes may then be compared in the automatic comparators that are available for this purpose. When all errors have been inspected one will usually produce a third copy of the tape, with the aid of automatic copying machines. In this copying process it is convenient also to include the library routines from the library tapes that are kept in the library. The symbols to be found on the final order tape will then group themselves as follows:

Main addresses, corresponding to the first order of each subroutine.

Main program.

Specifications symbols for the input routine. The input routine, copied from the library tape.

Specification symbols for the integration subroutine. The integration subroutine (copied from the library).

Auxiliary routine.

Specification symbols for the sine function subroutine. The sine function subroutine (copied from the library).

Specification symbols for the exponential function subroutine. The exponential function subroutine (copied from the library).

Specification symbols for the output subroutine. The output subroutine (copied from the library).

(A)

'Start program' combination. Numerical data, punched decimally.

(B)

Specification symbols for the check subroutine. The check subroutine (copied from the library).

(C)

f) Now the program is ready for a test run. It is therefore placed in the queue of experimental programs waiting for being tried. On the Edsac experimental programs have priority before production programs, but each experimental execution is given only a few minutes of machine time. This rule may of course be enforced only because the users have such a rich collection of check programs available. In a test of the program that has been described above there would be time for three attempts:

Normal execution, without check routine. In this mode the tape has to be placed in the reader of the machine and one has to press the start button. The input of the program will probably take less than a minute. It may happen that the machine stops before the program tape has been read entirely. In this case one will write a mark on the tape at the spot where it stops, and then use one of the so-called ‘post mortem’ check program. These may be used without preparations and will provide that the function letters or the addresses of the orders that are held in the store are printed. Several hundred orders may in this way be printed within a minute. This material will usually be sufficient to determine why the input of the program has stopped. If the program is read entirely and starts execution it will often by its behaviour indicate whether there it has errors. If it comes to a stop one will from the cathode ray tubes read the address of the order that has come to the stop. Under these circumstances one will make use of the check routine that one has prepared. This is done by stopping the reading of the tape at the point shown in the survey of the tape on page 24 as (A). Then the tape must be manually moved so as to read from (B) to (C) and thus place the check program in the store, and thereafter again moved as as to continue reading from (A). The information delivered by the check program may then be studied at leisure, without occupying the machine. Most of it will provide a check that much of what has been planned has also been put properly into effect. However, it turns out that a careful check of the output from a check routine or a ‘post mortem’ almost always will reveal where the error in the program has to be found.

In this connection there is occasion to discuss a detail that gives rise to some disagreement. The issue is the correspondence between the external and the internal code. It has already been stressed that this correspondence in the Edsac is ambiguous. Two orders that are identical in the internal code may appear differently in the external code. It is clear that in comparing the information supplied by a ‘post mortem’ program to the written external code one has to transform the latter to the internal code (or rather, to the external code used in the output from the ‘post mortem’ program). This may appear to be a defect in the coding system of the Edsac. In the author’s experience this is rather an advantage, however. The difficulty in the form of check at hand, proof reading, is indeed that one gets used to seeing the wrong thing, without reacting. When all of a sudden the same program appears in the internal code, one often notices errors that so far have passed unnoticed by habit. The demand that there should be a close correspondence between the external and the internal code arises from the assumption that program checks are made by direct comparison between the written code and the contents of the store. Such a comparison appears to this author to be so time consuming and tiring that it definitely cannot be adopted for general use.

For additional emphasis it should be noted that the Edsac allows very convenient reading of the contents of the store and of the control and arithmetic registers. In fact, this reading is quite convenient since the machine is serial and therefore allows direct display of the contents by cathode ray tubes. A parallel machine normally will not allow a similar display.

As conclusion of this account of the coding practice used with the Edsac a survey of the conditions assumed by the system will be given. It is already clear that the essential foundation is the library of subroutines. If this is to constitute a real support of the coding it has to build on a certain standardization. In fact it is not sufficient that the various users agree to put the subroutines, that they have occasion to write, into a common pool. In order to make a subroutine useful to others it is necessary that it is edited and provided with explanations in a uniform style. The point is that a program without comments is virtually unreadable to an outsider. An effective library therefore assumes office personnel who can do the editing and duplicating the subroutines.

The effectiveness of a library cannot be greatly dependent on the machine to which it belongs. Only one detail seems to be of some importance: the elementary input order. Curiously it seems that the arrangement of the library will be easier the simpler the effect of this order. In the Edsac a single order cannot read more than a single row on the tape. This makes it possible to process each row immediately. In many machines a whole word, *i.e.* several rows, will be read at a time. This probably always will make the input program more complicated, without corresponding advantages.

As an alternative to using a special order input routine for the input of programs, it is suggested in several places, *e.g.* at Harvard, that instead a special device, which may perform the necessary code transformations, is built. For example it may be mentioned that for the machine Harvard Mark IV a special coding machine has been built. This machine is equipped with several complicated mechanisms. This solution does not seem to present any significant advantages, however. The point of a comparison is that the use of an input routine corresponds to using the electronic computer itself as coding machine. The flexibility offered by the computer obviously cannot be built into any simpler machine. It might still appear that this would waste the time of the computer, an argument which also does not hold. The decisive time in input is spent on the reading itself. If punched tape is used as input medium the highest speed of input is hardly more than 200 symbols per second. A machine such as Besk may execute many elementary orders in 0.005 seconds, however. Consequently no extra time is used, even if several orders have to be executed in processing each row of the tape.

As the last point of this account of the Edsac system, a short description of the available auxiliary equipment will be given. In the Edsac coding room is available: 1) Three manually controlled tape punches. 2) Three tape duplicators. One of them is quite simple. Another one is a combined duplicator and manual punch. In this machine one may combine written symbols with symbols that are available on tape. This machine may also be used to skip the erase symbol. This has holes in all five positions and thus will appear wherever any other hole combinations have been overpunched with it. Thus erroneous symbols may be removed from a tape by first overpunching them with the erase symbols and then using 'copying with skip of erase'. The third duplicator has two reading mechanisms and may be used to merge two tapes into one. The machine will duplicate one of the tapes until it meets a hole combination that has been preset on manual switches. It then changes to duplicating from the other tape, until it meets another preset hole combination, etc. 3) Three tape comparators. Each machine has two read mechanisms. Two tapes that have been placed in the readers will run forward as long they carry the same the symbols and stop as soon as the two symbols differ.

When preparing tapes for the Edsac the need to proof read printed symbols is avoided. All checks are done in a comparison of two tapes. This is natural in a system in which most symbols in a program derive from library routines that have been copied automatically from standard tapes.

As the last issue of this section the organization around the coders will be taken up. The essential requirements of the organization are that the work of the coders is facilitated, while at the same time a good utilization of the machine is assured. These two requirements are in reality almost coincident, since the utilization of the machine rests on the coders' quick production of correct programs. The condition that the organization is satisfactory is therefore that effective aids to diagnosing coding errors are worked out. The point is that the chief difficulty in the practical utilization of an electronic computer is the elimination of elementary errors. A library offering checking routines and post mortem programs seems to solve this problem.

In this context it should also be stressed that the organization cannot assume that the users of the machine are specialists who have the machine as their exclusive concern. In many cases the reality probably is that really satisfactory solutions of special problems will not be achieved unless those who are interested in the problems code them themselves. Therefore the organization must aim at allowing outsiders to get familiar with the use of the machine.

Technical maintenance and fault handling

Experience shows that an electronic computer will not work for many days before it starts to fail. It is therefore necessary to make special precautions so as to protect the computed results from errors and so as to facilitate the work of finding and correcting the errors that occur. There are two problems to be considered: 1) How to make sure that the results delivered by the machine are flawless, and 2) how to prevent that the machine fails at inconvenient moments?

For checking the work of the machine two different methods are proposed: a) The machine is provided with automatic check equipment, which provides checks of all internal operations. An example of a machine that is equipped with this sort of check system is Univac. In this machine most of the arithmetic unit is built in two copies, for mutual check. The contents of the working store are checked at brief intervals with the aid of special check pulses, that occur with a frequency of one per each six information pulses. The check consists in that within each complex of seven pulse times there must occur an odd number of pulses.

An automatic check system of this kind undoubtedly will provide a practically perfect assurance against machine errors. As an additional advantage, each error is immediately localized in the machine. However, the cost of this kind of assurance is that the machine becomes enormous more expensive. Further, there will occur more errors in the apparatus as a whole, since it contains extra hardware. Another shortcoming is that this kind of check system never will contribute to the detection of program errors.

b) The alternative to automatic built-in checking is the use of mathematical checks in the programs. This practice agrees with the view that all complicated functions in the machine have to be performed by subroutines, as described in chapter 6. The basic attitude is that the most effective method to reduce the number of errors in the hardware is to reduce its bulk. The efficiency of mathematical checks depends highly on which problems are solved. If large amounts of empirical numerical material are involved it may cause much extra work to introduce numerical checks. But in physical or mathematical problems it is often possible to introduce simple but effective mathematical checks. Often these will be of such a kind that programming errors to some extent will be revealed.

The check of input and output raises special difficulties. However, input may be checked if suitable check numbers, *e.g.* check sums, are included among the useful data. Upon input the program then immediately should perform the check operation. Output is more difficult to check. But it is possible to indicate errors by using a suitable logical arrangement. For example the output from the Edsac is produced by a five-hole teletype punch. Each symbol on the tape then gives 32 possibilities. An effective check is achieved by a suitable choice of the code for these 32 possibilities. The decisive property of this code is that the digits from 0 to 9 all are represented by symbols having just two holes. When the hardware fails it is likely that one hole is omitted from or is added to the correct pattern. However, such a failure will never convert a decimal digit into another. Instead a symbol, such as $, ;, %, or (, will be printed, which immediately reveals an error. Only in the case of a double error, which erroneously adds one hole and omits another one, will a wrong digit be printed.

Preventive maintenance

The first preventive maintenance step has to be taken already during the planning of mechanical layout of the machine, since it will be decisive to the maintenance that all parts of the machine are readily accessible. A further step in the same direction has been taken in many machines, by building the machine from smaller hardware units that easily may be taken out of the machine, so-called plug-in units. Such units yield substantial advantages: 1) The machine may often be built from a large numbers of identical components of this type. 2) Faults in the machine may quickly be remedied, provided only there are extra components available. 3) The testing of the hardware may be done on those extra units that are not in active use. Merely at the cost of a small amount of extra hardware it thus becomes possible to carry out a continued cycle of maintenance, without using the time of the machine itself.

The faults that appear in electronic hardware are of two types: a) Spontaneous defects, *e.g.* open heaters in tubes. b) Faults that arise from the steady deterioration of the properties of the components, *e.g.* the reduction of the cathode emission of tubes.

Spontaneous defects probably cannot be entirely avoided, but their frequency may be reduced by the use of the best possible quality of hardware components and by deliberate underloading of it. The faults that arise from spontaneous defects may usually be detected and corrected fairly easily.

The most difficult faults in electronic computers arise from the age effects. The gradual deterioration of a component will often have the effect that at a certain stage it will fail a certain small fraction of the times it has to function. These failures will induce faults that cannot be made to repeat themselves, which makes it very difficult to localize them. The best technique for overcoming this type of problem has so far been the use of marginal testing. The principle is that a component which is working near the limit of the range where it functions reliably may be brought to fail more frequently if its working conditions are changed. The machine is therefore equipped with suitable switches that allow each part of it to be subjected to slightly changed conditions. Typically certain supply voltages are made adjustable within a certain range around their normal values. The machine is then considered to be in a workable condition only when it executes certain test programs without failure, not only when working under normal conditions, but also when all of its units at the same time are adjusted to a less favourable condition. The alternative to this kind of testing is the separate test, mentioned above, of each unit of the machine.

The detection and diagnosis of faults in an electronic computer may be helped enormously by employing the fact that the machine is a general purpose device. Thus it may be used, to a large extent, to indicate and localize its own faults. For this purpose special test programs are constructed for the machine. This allows wide variations, since a single program may test all of the functions in turn, or may concentrate upon a single function. A general requirement of test programs is that they must deliver useful information about each fault that is detected.

Proposal for a Danish nomenclature

The present proposal for a Danish nomenclature is an attempt to avoid that a certain linguistic development, which unfortunately is already far advanced in several scientific fields, should also become dominant within the field of electronic computers. It is well known how in many cases technical terms have been taken over from foreign languages directly, without translation. In this way the reading of foreign technical literature may become facilitated, and the need to decide upon a suitable Danish term is avoided. However, these advantages are overshadowed by the awkwardness that anyone who has any sensitivity to the individuality of languages must feel at using such a language mixture.

The field of electronic computers so far has been so little cultivated in Denmark that there appears to be a reasonable hope that this unfortunate linguistic development may be avoided. However, for this to succeed a certain determination among those who are concerned is necessary. May the present list of words contribute to supporting such determination.

List of words

Danish	*English*	*Remarks*
Arbejdslager	Main store	
Bånd	Tape	Compounds: *Hulbånd* and *magnetisk bånd*
Binal	Binary digit	Analogue: *Decimal*
Hjælpelager	Auxiliary store	
Hovedrute	Master routine	
Hulbånd	Punched tape	
Huller	Punch	
Indlæsning	Input	
Indre kode		The explicit distinction between the internal and the external code is, as far as I know, original in the present discussion
Kode	Code	See also *indre* and *ydre kode*
Kontrol	Control	The control is the organ that controls the work of the machine. The designation is perhaps ill chosen, since a confusion with organs that test the correctness of results may arise
Lager	Store, memory	The alternative, *hukommelse,* is long and suggests irrelevant associations
Løkke	Loop	
Ord	Word	
Overløb	Overflow	
Plads	Location	Compound: *Lagerplads*
Program	Program	
Rute	Routine, sequence	
Seriel	Serial	
Siderute	Subroutine	
Sinkelinie	Delay line	
Spor	Track	
Spring	Transfer of control	
Stødpude	Buffer	
Tromle	Drum	Compound:*Magnetisk tromle*
Udskrivning	Output	
Ydre kode		See *indre kode*