input register and transfer 8 bits by way of input J into the input register. Input J is connected to signal source by plugboard.

250 Transfer the contents of register 2 to the output register by way of L: reset the output register, transfer the contents of register 2 to the output register by way of the plugboard and entry terminal L of the output register.

251 Transfer contents of register 2 to the output register by way of M: reset the output register, transfer the contents of register 2 to the output register by way of the plugboard and entry terminal M of the output register.

Group 3
The execution time for orders in this group is 10 microseconds except for order 34x. In the jump orders the eight low-order bits of A may be overlaid with the contents of BX. When the low-order digit of the code is designated by x, x can have the value of 0, 1, 2, 3, 4, 5, 6, or 7.

310 Jump to (a): unconditional program transfer to the address (a) specified in the instruction register.

317 Jump to (a) and stop: unconditional program transfer to the address (a) specified in the instruction register and stop before the execution the new instruction.

311 Jump if contents of register 1 =0: conditional program transfer to the address (a) specified in the instruction register if the contents of register 1 =0. If contents of register 1 #0, go to the next instruction.

312 Jump if contents of register 2 =0: conditional program transfer to the address (a) specified in the instruction register if the contents of register 2 =0. If the contents of register 2 #0, proceed to the next instruction.

313 Jump if contents of counter P =0: conditional program transfer to address (a) specified in the instruction register if the contents of counter P =0. If contents of counter P #0, proceed to the next instruction.

314 Jump if contents of carry storage =1: conditional program transfer to address (a) specified in the instruction register if the contents of carry storage =1. If contents of carry storage #0, proceed to next instruction.

315 Jump if contents of borrow storage =1: conditional program transfer to address (a) specified in the instruction register if contents of borrow storage =1. If contents of borrow storage #0, proceed to the next instruction.

32x Jump if indicator latch Jx is in “one” condition: conditional program transfer to address (a) in the instruction register if the indicator latch is in condition “one.” If indicator latch is in condition “zero” proceed to next instruction.

33x Set indicator latch x to “one” condition.

37x Reset indicator latch x to “zero” condition.

34x Execute plugboard controlled instruction x.

35x Sense on breakpoint switch x. The breakpoint switch is not shown in the circuit diagram. If breakpoint switch x is in “off” position, proceed to the next instruction. If the breakpoint switch x is in “transfer” position use the address (a) specified in the instruction register for the address of the next instruction. If the breakpoint switch x is in “stop” position, prepare to use the address (a) specified in the instruction register for the address of the next instruction and stop.

---

A Universal Computer Language Translator

R. B. Bonney
Nonmember AIEE

Can Univac talk to Bizmac? As more and more electronic data-processing systems are put into operation, the need for rapid and efficient information interchange becomes important. Many organizations have evolved data-processing systems using a combination of different computer types, together with a variety of input and output equipment. The use of different computers in a single organization does not necessarily represent poor planning, since the requirements of one group may dictate the use of a particular type of computer which is inadequate for the requirements of a different group.

To achieve efficient utilization of an electronic data-processing system, it is often desirable to transfer output data from one computer to a different computer for additional operations.

The need for interchange between different organizations is also beginning to develop, and can be expected to increase as electronic data-processing spreads. Acceptance of payroll deduction reports on International Business Machines Corporation (IBM) 705 magnetic tapes by the Treasury Department and the Social Security Agency are typical examples.

Transformation from one form of data to another is another common requirement. Conversion from magnetic tape to punched paper Teletype tape is representative of a wide variety of such requirements.

Unfortunately, the designers of electronic computers have made very little attempt at standardization. In nearly all cases, the recording format and coding used by the various computer manufacturers are sufficiently different to prevent direct interchange of data. Undoubtedly, the very rapid development and growth of the computer industry have been major factors in the lack of standardization.

The commonly used method of accomplishing data translation between computers up to now has been by means of punched cards, and in some cases punched paper tape. Translation by these methods is seriously hampered by the speeds of operation. Some special-purpose high-speed translation equipment has been built for direct translation of data on magnetic tape. However, because of the wide variation in requirements, the equipment developed for a specific application ordinarily cannot be used in a different application without modification or complete redesign.

Realization of the need for "universal" translation capability resulted from work on data-handling systems for scientific test data. For this purpose, a computer format control buffer was developed to accept unsynchronized, continuous digital input data which are to be converted to magnetic tape, in a format directly usable by a computer.

In discussing other applications of this equipment, it soon became evident that no two applications were alike, and that too much re-engineering would be required to adapt the equipment to each new situation. It was, therefore, decided that an attempt should be made to develop a system with the flexibility necessary to meet all commonly encountered translation requirements without becoming overcomplex and economically impractical. The Computer Language Translator is the result.

In considering all possible applications, there is a wide variety of requirements...
Bonney—A Universal Computer Language Translator

which must be fulfilled by the translator equipment. A major undertaking in connection with this development has been the assembly of the myriad of details required for each computer format and for the numerous input and output devices which are expected to operate with the Translator. Although the equipment manufacturers have generally cooperated admirably, the problems in sorting out and verifying the necessary details have been considerable. The available literature is helpful but ordinarily does not give sufficient information, and it is difficult to locate the individual who can provide all of the answers.

Another problem has been to incorporate the necessary intelligence into the equipment without the Translator growing into a full-fledged digital computer.

The reader may ask, “Why not use one of the smaller general-purpose digital computers rather than develop new equipment specifically for translation?” There are a number of answers to this question:

1. All of the smaller computers currently available use magnetic drum memory, and as a result their speed of conversion would be much too low.

2. A magnetic drum is usually not suitable for delivering synchronous output data at a rate different from the input. This is one of the most fundamental translation requirements.

3. A large part of the translation process is involved with control of the data input and output devices. A general-purpose computer cannot provide these control features.

The required flexibility has been provided by means of the “building block” modular technique. Thus, by providing the proper combination of modules, the needs of a particular data processing system can be fulfilled without providing an overcomplicated system, containing excess features and equipment.

The design has been worked out around a foundation building block called the universal format control. This portion of the system is common to all applications. It contains a buffer memory, operator’s controls, counters, and power supplies required for most operations.

Functions which are peculiar to a particular type of data input or output device are treated separately through the use of input and output control units. Ordinarily a separate control unit is required for each different type of data to be handled. This works out to be feasible from an equipment cost standpoint, since most input or output controls represent less than five % of the cost of the central universal format control.

Fig. 1 is a block diagram of the basic system. For multiple function systems, the universal format control can be shared between several input and output control units through the use of patchboards as shown in Fig. 2.

The operation of the Computer Language Translator can be divided into the two general categories of format conversion and code conversion.

Format conversion is concerned with the transformation of the physical characteristics of the input data to the required output form. Format conversion may involve radically different forms of input and output such as the translation of punched cards or punched tape to magnetic tape. On the other hand, the format conversion may require the translation of one magnetic tape to another magnetic tape with different recording speed and geometry.

Code conversion may or may not be required, depending on the application. Both character and word coding can be handled. Following the building block concept, the code converters are provided as auxiliary equipment when required, rather than as a part of the basic system.

In character conversion, the coding of each character is changed, as for example binary coded decimal to excess-three.

Word conversion involves reordering the characters within a word and transposition of the sign character where necessary.

Modification of the word order within a data block would be helpful in some cases. However, this feature has not been included in the interest of simplification. Deletion of certain words or characters from each block can be handled so long as the order remains fixed for any one operation.

In the normal operating mode, data is transferred from the input device to the buffer memory with the output equipment stopped. The amount of data entered can be arbitrary, up to the capacity of the buffer memory (1,092 or 2,184 characters). On completion of loading, the input device is stopped, the output device is started, and the information is transferred from the buffer memory to the output.

In general, it is not practical to operate with the input and output devices running
simultaneously. Unless the input and output rates are synchronized, the buffer memory will eventually overflow or empty. Since the input and output rates are normally different, the control logic is greatly simplified by using separate load and unload cycles.

In cases involving word conversion and for some other conversion operations, a recirculation cycle is introduced between the input and output cycles. This is illustrated in Fig. 1. On completion of transferring the input data to the memory, the recirculation cycle is initiated. During this cycle, the data is unloaded to the word converter one word at a time. The word converter is in turn unloaded, through the character converter, back into the buffer memory. At the completion of the recirculation cycle, all of the data will be in the required output coding. The memory can then be unloaded as a complete block at the rate required by the output device.

Insofar as possible, the equipment has been designed to eliminate the need for correlation of the input and output data formats.

The output word structure and block length can be entirely different from the word and block structure of the input data. In cases where fixed word length or block length output is required, zeros or other fill characters can be inserted into each word, and zeros or fill characters will automatically fill out to the proper block length. Several short input blocks can be compressed into a single output block.

Elimination of dependence between input and output format is also achieved by entering control characters into the memory. For example, when operating with variable input block length, an "end of data" character is entered into the buffer memory at the end of the input cycle. This special character is detected at the memory output and used to control the output cycle. The use of control characters thus eliminates the need for counting circuits to determine the output block length. The special characters, in most cases, do not appear in the output data.

It is doubtful that a detailed functional block diagram of the Computer Language Translator would be of interest to most readers. The function of each component part may not be obvious from the block diagram without a step-by-step detailed description which is beyond the scope of this paper. It is felt rather that a brief listing of the functions and characteristics of the system will assist in illustrating the design philosophy.

As stated previously, all requirements peculiar to a particular input or output device are handled in the associated input or output control units. This results in much greater flexibility, since the designs for new types of control units can be completed individually without involving the majority of the system.

Input to the system can be in the form of: magnetic tape, punched paper tape, punched cards, analog to digital converter register, or other "on-line" source.

Output can be in the form of: magnetic tape, punched paper tape, punched cards, on-line registers for use by digital to analog converters, or other similar applications.

A rather formidable list of requirements for input and output control units results if all likely translator applications are considered. The situation is further complicated by various optional equipment combinations which may be desirable for various reasons. As an example, magnetic-tape data can be accepted directly from an IBM 727 tape unit, or 727 tapes can be played into the system using an Ampex FR-307 or FR-407 digital tape transport which is an integral part of the Translator. A somewhat different control unit is required for each case.

To date approximately 25 input and output control units have been designed or are in process. These include control units to handle magnetic tapes for IBM, ElectroData, and Remington Rand computers. Also included are control units for punched paper tape, IBM punched cards, and line printers.

By using the central portion of the system to fulfill more than one operational requirement, the cost per function decreases as additional operations are added. The central universal format control represents between 50 and 75% of the total equipment for most systems. The universal format control is designed to operate with any input or output control unit. Thus, the cost of adding an additional function to the system is essentially the cost of the input or output control unit plus any associated patchboard wiring.

To date Electronic Engineering Company has been reasonably successful in isolating the universal format control from special input or output data requirements. Anticipating the need for
considerable flexibility, circuits needed to adapt the universal format control to different translation applications have been brought out to external terminals. By providing the necessary control signals to these terminals, any requirement within the basic system capability can be handled. In some cases, auxiliary equipment is designed to provide the necessary controls when they are not directly obtainable from the data input or output equipment.

All normal operator's controls are located on a panel housed in the universal format control. For magnetic tape input, provision is made for positioning the tape both automatically and manually in either forward or reverse direction. Indicating counters keep track of the block or column position of the tape.

The actual translation operation can also be controlled automatically or manually.

Other controls and indicators are associated with marginal testing and parity-error control circuits. All controls are interlocked to prevent improper operation.

Character parity is checked at the input before entering the buffer memory and also at the output of the memory. A third parity-check circuit is also provided for output parity checking when possible, as for example with the IBM 729 tape unit. At the operator's option, the controls can be set to automatically stop the operation if a parity error is detected.

Longitudinal (row) parity, if used, can also be checked by means of an optional module.

Marginal operation checks are provided for both the buffer memory and the d-c supplies for the balance of the system.

A common translation requirement is for off-line conversion from magnetic tape to punched cards or line printer and from punched cards to magnetic tape. The multipurpose features of the Computer Language Translator work to advantage here, since the basic system can be adapted to handle these off-line functions in addition to tape-to-tape or other requirements.

As anyone knows who has considered the problem, existing card-handling equipment and line printers are not easily adapted to magnetic tape. This is because the most commonly used equipment handles the data on a row-by-row rather than a column-by-column basis. Some column-by-column card-handling equipment is available, but it is too slow in most cases. A number of high-speed column-by-column readers are under development, but as yet are not generally available. The straightforward approach to handling row-by-row cards is to provide a complete card size or printer line buffer memory which can be unloaded or loaded on either a row or column basis. Because of the number of parallel outputs involved (120 in the case of a line printer), this type of memory would be a relatively expensive addition to the translator system.

It is possible to utilize the buffer memory in the basic translator to handle cards and line printing by operating the memory in a manner equivalent to a magnetic drum. Using this approach, the stored information is recirculated in the same manner as discussed previously for word conversion.

Fig. 3 is a block diagram illustrating the method used for punching cards or controlling a line printer such as the IBM 709.

Data corresponding to one card row or one line is first loaded into the memory from the input equipment. The memory is then recirculated 12 times in synchronization with the 12 card row positions. During each circulation cycle, a decoder detects characters which correspond to the row position and are to be punched or printed.

A column counter, and associated setup patchboard, sets up the proper punch magnet or print magnet. Since this same counter controls unloading and reloading the memory, the required synchronization is maintained between the memory and the punch or printer columns. A similar approach is used for reading punched cards. Each row of the card is scanned sequentially by the column counter. A card-to-tape coder enters the proper character into the memory when a punch is detected.

Card reader verification can be handled by double reading. This necessitates a reader with two reading stations. Verification is done during the second reading by comparing the reader output against the decoded data entered into the memory during the first reading.

Card punching can be verified in a similar manner. In this case, the punched card is read following the punching cycle. The information from the reader is compared to the decoded output from the memory.

The translator is designed around a combination of plug-in vacuum tube circuits, diode logic circuits, transistor-driven ferrite cores, and in some cases tape wound cores. Some consideration was given to eliminating vacuum tubes entirely. However, the idea was not carried out because of the considerable amount of design knowledge and proven reliability of the vacuum tube plug-ins. These same components have been thoroughly proven by use in many other similar applications. On the other hand the transistor-driven ferrite core memory was used since this unit was originally developed using transistors. Tape wound cores are being used for the card and line printer column counter because of the relatively large linear count and number of output circuits involved.

In general, d-c logic is employed. Conservative design policies have been established in the interest of achieving high reliability. Active circuits are isolated from external loading. Definite rules for interconnected logic circuits are followed.

Standards are maintained for component parts and workmanship which are somewhat higher than are often found in commercial electronic equipment. Transformers, for example, are hermetically sealed. Other components are of top quality. It is sincerely believed, however, that the modest cost for high-quality equipment is more than offset by higher reliability and decreased maintenance requirements.

The basic system is housed in two equipment racks which are each 2 feet square and 7 feet high. One or more additional racks house input and output control units, patch-boards, auxiliary power supplies and, in some cases, magnetic tape transports.

Fig. 4 shows a typical translator system. This system was demonstrated at the Western Joint Computer Conference. The input to the system comes from punched paper Teletype tape or from ElectroData 205 magnetic tape. The output is in IBM 727 format. The input tape unit is the ElectroData transport used with the 205 Datatron computer. The output tape unit is an Ampex FR-307 tape recorder with IBM head dimensions.

Until more varied application experience has been obtained, it will be difficult to assess the success or failure which has been achieved in developing a truly universal Computer Language Translator. Experience to date, based on actual applications and discussions with many potential users, has been very encouraging.
A Computer Oriented Toward Spatial Problems

S. H. UNGER
NONMEMBER AIEE

The stored-program digital computer has been in existence for about a decade and has proven itself to be a powerful and versatile instrument. Roughly speaking, any solvable problem can be solved by a digital computer. At many tasks, such as the solution of systems of linear equations, these machines are thousands of times as fast as human beings.

However, there are certain tasks, which might be termed spatial problems, at which digital computers are relatively inept. For example, it seems to be extremely difficult to write chess playing programs that will enable a computer to compete successfully with a capable human opponent. Pattern recognition is another area in which present-day machines cannot match the performance of their designers. References 2, 3, and 4 describe some important and interesting work done in this field.

The difficulty in such cases appears to be that these machines can actively cope with only a small amount of information at any one time. This circumstance is aptly illustrated by the title of an article by A. L. Samuel, “Computing Bit by Bit,” which appeared in the computer issue of the Proceedings, Institute of Radio Engineers. It appears that efficient handling of problems of the type previously mentioned cannot be accomplished without some form of parallel action.

An important implication of this argument is that machines of greater complexity are needed. This, however should not be regarded as an insurmountable barrier, since it is likely that great advances will be made in the components field during the next decade.

The principal objective of the research reported on here is to learn how to build machines that can efficiently solve problems not well suited to solution by conventional digital computers.

The approach in this paper is to attempt to describe a computer which is inherently well matched to one such problem (pattern detection), hoping that such a computer, probably in an improved version, will then perform well when faced with a much larger class of problems.

General Structure of the Computer

Fig. 1 illustrates the form of the proposed machine. It consists of a master control and a rectangular array of modules. Each module communicates with its four immediate neighbors and receives orders from the master control. The master control cannot address the modules individually, but issues general orders which go to all of the modules.

A module consists of a one-bit accumulator, a small amount of random access memory, perhaps six bits in one-bit words, and some associated logic. Inputs to each module come from the master control and from the accumulators of the modules above, below, to the left, and to the right of it. It will also be assumed that an input signal may be fed directly to each accumulator from outside the machine.

The master control includes a random access memory for storing instructions, a clock, and decoding circuits. It acts like the operation decoding section of a conventional digital computer, reading out instructions from memory in sequence, decoding them, and sending appropriate control voltages out on a set of buses feeding the modules.

A logical adder (or-gate) with n inputs from each module accumulator tells the master control if all of the accumulators have zeros in them, and thereby makes possible the use of a transfer on zero order. This instruction, analogous to the conditional transfer orders used in ordinary digital computers, tells the master control to skip to the instruction addressed by the transfer zero order if there are no ones in any of the module accumulators.

It is the only decision order used in this computer.

Order Structure

The order structure is summarized in Table 1. The transfer on zero order (tr z x) has just been described. In addition there is an unconditional transfer (tr x) which simply orders the execution of the instruction at address x.

As was mentioned before, each module has a small number of individually addressable one-bit memory elements. Within a module these registers will be referred to as a, b, c, etc. The instruction store b (st b) tells each module to store the contents of its accumulator in the bth memory register without altering the accumulator contents. Similarly write b (wr b) orders the contents of each b register to be written into the accumulator (without changing the contents of b).

Information can be transmitted directly from one module to another by means of a shift instruction. A shift right (sR) order for example, causes the contents of each accumulator to be transferred to the accumulator of the module to the right of it. Shift left (sL), shift up (sU) and shift down (sD) have analogous meanings. When shift orders are used assume the matrix is bordered by modules having zeros in their accumulators.

The invert order (in) causes the contents of each accumulator to be comple-

S. H. Unger is with Bell Telephone Laboratories, Inc., Whippany, N. J.

The idea of considering switching circuits in n-dimensions came to the author through the work of Professor D. A. Huffman of M.I.T. Thanks are due to C. Y. Lee and M. C. Paull of the Bell Telephone Laboratories for many interesting conversations which contributed directly and indirectly to the work presented here, and for their advice regarding the preparation of this report.

From the collection of the Computer History Museum (www.computerhistory.org)