The interconnection of the digital components could be accomplished with multiconductor patching or switching within the digital computer itself. The actual computer setup time for interconnection of units would be approximately the same as a conventional machine.

The information to be stored on the drum would best be handled with punched cards. Functions such as sines or cosines could be calculated and punched on cards using standard digital computers. Empirical function could be either punched into cards using a manual keyboard, or if desired, curve followers with digital outputs could be used to encode the desired information automatically. Once the set of data has been placed on the cards the information is read into the drum using well known techniques. If standard equipment were used each function would require approximately 10 minutes to be read into the drum. Once the function is on cards it could be stored and would be available at any future time for use on the machine.

Conclusions

The function table computer together with a suitable analogue computer would be a machine capable of solving many types of problems faster than is now possible with analogue computers. The accuracy of the proposed system would be comparable with present computers. Programming is simplified because of the ease of computing with any function of a variable. The combined computer should be easier to maintain because of the absence of servomultipliers and resolvers. Checking the operation of the digital function table may be accomplished by standard analogue techniques. At present the machine is feasible because it uses available equipment and techniques. Future development in components may reduce the cost of the computer and enhance its usefulness.

An Experimental Monitoring Routine for the IBM 705

H. V. MEEK

THERE is a pressing need for aid in the knotty business of checking a code for a large digital computer. What better instrument is to be used than the computer itself? Because an automonitoring feature is absent from the circuitry of most computers, the monitoring operation should be programmed and will furnish an effective means of detecting many coding errors.

A routine for the International Business Machines Corporation (IBM) *705* has been prepared which monitors the instructions of a code being tested, and gives a complete history of the computer action as a result of that code.

Before the format of the output of the monitoring routine is discussed in detail, a brief description of the *705* internal nature should be given. Also, some of the questions which arose during the monitor planning phase and the decisions which were made should be mentioned.

The *705* is a high-speed stored-program electronic data-processing machine. Its main memory is either 20,000 or 40,000 character positions of magnetic core storage. Each character consists of seven binary digits; a 4-bit "numeric" part, a 2-bit "zone" part, and one check bit, and may be a decimal digit, a letter, a punctuation mark, or a special symbol. A 256-position accumulator, and 15 auxiliary storage units which together comprise 256 positions, provide temporary working storage for arithmetic and logical operations. Instructions have five characters each: one operation-code character and a 4-decimal digit address part. The thousands position of the address part must be appropriately zoned to specify a location with address greater than 9999. If one of the 15 auxiliary storage units is to be specified, the tens and hundreds positions of the address part are zoned so that the four zone bits together form a binary number equal to the address of the desired auxiliary storage. There is no fixed word length in the *705;* the length of an operand depends on the current condition of the specified accumulator or storage, the type of instruction being executed, and/or the characters of the operand and its adjacent fields.

Now then, what exactly should the monitor tell about a code? It was decided that each executed instruction, its location, its interpretation, the operand it calls for, and the results it produces would be given. The instruction interpretation consists of a 3-character mnemonic operation symbol, a 5-decimal digit address part, and a 2-digit reference to the accumulator or auxiliary storage unit. Each operand, including its length and sign, each change in main memory content, and the length, content, and sign of the accumulator or an auxiliary storage unit after each reference will be noted alongside the corresponding instruction. Flags will call attention to coding irregularities, such as specifying an auxiliary storage unit for a "multiply" instruction, or allowing the character to the left of a field stored to be signed. Also, before any monitoring commences, the initial length, sign, and contents of the accumulator and each auxiliary storage unit will be given.

Experience with the routine may show that some of the output is not useful enough to warrant the memory space and additional execution time required, and a subsequent version of the monitor may be written which does not include such features. Meanwhile, starting with the assumption that it will be easier to trim than to add, an effort was made to make the monitoring routine output a really complete record of the computer activity.

Paradoxically, a prime requirement of a monitoring routine is that it have the ability to execute sections of a code without monitoring and at full speed. There was no question that this feature would be included in the *705* monitoring routine, the problem was how to specify those sections. The method chosen is simply this: The coder prepares sequence cards, each specifying the location of the first

H. V. Meek is with the Hughes Aircraft Company, Culver City, Calif.

instruction of a sequence to be monitored and the location of the last instruction of the sequence; and the number of those times that the sequence is encountered it is to be monitored. As many as five different sequences may be specified for a single monitoring run.

Before monitoring begins, a table of the information from the sequence cards is automatically compiled in the computer, and the first instruction of each sequence is saved and replaced by a "transfer" instruction which will transfer control to an appropriate monitor entry point. When the table is completed, control is transferred to the first instruction, the location of which is given, along with the address of the unit selected for monitor output, on another card, of the code being tested. Thus, full-speed execution of the code takes place until a sequence to be monitored is encountered; then the contents and length of the accumulator and each auxiliary storage unit is given, and monitoring commences, continuing until the last instruction of the sequence has been executed. At that time, the accumulator is restored to its current condition with respect to the code being tested, the "transfer any" indicator and the check indicators, except for "sign" and "overflow," are off, control is returned to the code being tested, and full-speed unmonitored operation again takes place until another sequence, or the same sequence, if it is to be monitored more than once, is encountered.

Another very important question is: Should monitoring time, or monitoring routine memory requirements be minimized? The nature of the *705* (with its variable word length feature and somewhat complex instruction composition) requires that a great many operations be performed on a code to produce the desired information. Memory space is at a premium for, after all, the routine could give no service if no other code would fit into the memory to be tested, and its use must be minimized, even at the expense of slower monitoring.

Finally, what restrictions are to be placed on the coder who wishes to use the monitoring routine, and what rules must he follow? Naturally, the fewer the restrictions, the better. The less complex the rules, the better. Alteration switches and check indicator switches should be set as if for a normal, unmonitored run. A sequence of monitoring must neither begin nor end between a "compare" instruction and its associated "transfer on high" and "transfer on equal" instructions, nor between a "select" instruction and its associated input, output and "transfer on signal" instructions, nor begin between a "receive" instruction and its associated "transmit" instructions. The coder must be careful to specify for monitor output the use of a tape unit or printer which will not affect the operation of the code being tested. For example, if he uses a type-*760* control unit in his own code, he should not choose for monitor output a tape unit or printer which is controlled by that same *760*, since then the contents of the *760* storage would be changed after every instruction. However, monitor output to a tape unit controlled by a type-*777* tape record coordinator (TRC) is written so that the TRC storage is by-passed and its contents remain undisturbed. And, naturally, neither the code being tested nor the data it uses should overlap the monitor in the memory. The monitoring routine which has been machine tested has no provision for monitoring the instructions peculiar to *760* and *777* operations, but a subsequent version of the monitor will have that provision.

Because the user of the monitor has only a few restrictions, the writer had several. For one thing, there is no practical way to save the contents and sign of one or more of the auxiliary storage units, use the storages, and then restore them to their original conditions. Therefore, only the *00* accumulator is used to compile and edit the output information. Since it is possible that the code being tested has a "shorten" instruction designed to recover the remainder of a divi-

Table I

Operation Code	Mnemonic Symbol	Information Given in Monitor Output (See Legend)	Approximate Central Processing Unit Time in Milliseconds
A	NOP	(0)	25
B	SET	(0), (6), (7), (8)	30
C	SHR	(0), (9), (10), (11), (12)	33
D	LNG	(0), (9), (10), (11), (12)	33
E	RND	(0), (9), (10), (11), (12)	34
F	ST	(0), (4), (5), (13)	39
G	ADD	(0), (1) (2), (3), (6), (7), (8)	52
H	RAD	(0), (6), (7), (8)	32
I	TRA	(0)	26
J	HLT	(0)	Manual restart time
K	TRH	(0)	23
L	TRE	(0)	22
M	TRP	(0), (16)	28
N	TRZ	(0), (16)	28
O	TRS	(0)	28
P	SUB	(0), (1), (2), (3), (6), (7), (8)	52
Q	RSU	(0), (6), (7), (8)	32
R	WR	(0), (15)	20+execution time
S	RWW	(0)	28
T	SGN	(0), (1), (3), (4), (6), (7), (8)	60
U	RCV	(0)	19
V	MPY	(0), (1), (2), (3), (9), (10), (11), (12)	53
W	DIV	(0), (1), (2), (3), (9), (10), (11), (12)	53
X	NTR	(0), (6), (7), (8)	43
Y	RD	(0), (15)	20+execution time
Z	WRE	(0), (15)	20+execution time
1	TR	(0)	21
2	SEL	(0)	26
3	Control Instruction[1]	(0)	20+execution time
4	CMP	(0), (1), (3), (6), (8)	58
5	SPR	(0)	30
6	ADM	(0), (1), (2), (3), (4), (5), (14)	52
7	UNL	(0), (6), (7), (8)	31
8	LOD	(0), (6), (8)	31
9	TMT	(0)	19

Legend:

(0) location of instruction (5 decimal digits), mnemonic symbol, address part (5 decimal digits) accumulator or auxiliary storage unit (ASU) reference, and the uninterpreted instruction
(1) operand
(2) minus sign if operand is negative
(3) length of operand
(4) result in memory
(5) length of result in memory
(6) result in designated accumulator or ASU
(7) minus sign if result in designated accumulator or ASU is negative
(8) length of result in designated accumulator or ASU
(9) result in accumulator
(10) minus sign if result in accumulator is negative
(11) length of result in accumulator
(12) "*" if an ASU is designated
(13) "*" if the character to the left of the field "stored" was signed plus
(14) "S" or "U" according to whether the ADM is "signed" or "unsigned"
(15) "X" if reading is under control of Memory Address Counter II and TMT has been issued since the last RWW
(16) address of the last changed ASU if an ASU is designated

[1] The appropriate mnemonic symbol, determined by the address part of the control instruction, will be given.

sion, it is necessary that the starting point counter of the *00* accumulator remain unchanged by the monitor. This requires that the monitor have no "shorten," "lengthen," "round," "multiply," or "divide" instructions. One further restriction on the writer: The "sign" and "overflow" check indicators may never be turned on by the monitor, for that would interfere with the use of those indicators by the code being tested.

The instructions needed to produce the desired output and still satisfy the foregoing rules are, of course, more numerous than those needed if complete freedom in the use of the *705* had been possible. Between 4,000 and 5,000 memory locations are used by the monitor to store its 675 instructions and 400 characters of constants, and to provide 650 positions of temporary storage. The constants include a table of 192 characters which relates the operation codes to their corresponding mnemonic symbols. The large amount of temporary storage is necessary for concurrently saving the contents of the accumulator, determining the length of an operand or result, and compiling the output record. The monitor code is a relative code and may be assembled to operate in any part of the memory, provided the 192 table character location addresses all have the same thousands digit.

Table I lists each operation, the type of information that will appear in the history, and the approximate central processing unit time required to collect that information and write a record.

The Logical Design of a Digital Computer for a Large-Scale Real-Time Application

M. M. ASTRAHAN B. HOUSMAN J. F. JACOBS
R. P. MAYER W. H. THOMAS

THE Lincoln Laboratory and International Business Machines Corporation (IBM) have, over the past 3 years, worked out the design for a new digital computer which is the central component of a large-scale real-time system. In this system, data from a large number of sources are fed automatically into the computer where they are processed under programmed control. A complete compilation of the real-time situation is compiled by the computer and presented to operators by means of a special display system. The computer automatically generates control commands for the external environment in response to corrections and command information fed into it by the operators.

The purpose of this paper is to describe the performance criteria of the computer in general terms, and to present some of the outstanding features of the design. These features are necessary in the computer because of its particular real-time application which requires a greater emphasis to be placed on reliability, speed, capacity, and flexibility than is usual in scientific or commercial applications.

M. M. ASTRAHAN, B. HOUSMAN, and W. H. THOMAS are with the International Business Machines Corporation, Poughkeepsie, N. Y.; J. F. JACOBS and R. P MAYER are with the Lincoln Laboratory of the Massachusetts Institute of Technology, Lexington, Mass.

Disregarding its special features for the moment, the computer is a large-scale general-purpose single-address parallel digital computer with a 32-bit word length. A high-speed magnetic core memory is provided which contains 270,336 bits of storage arranged in two banks of 33 planes, each plane consisting of a 64 by 64 core matrix. The 33d plane is used for parity check bits. In addition to its buffer drums for communicating with the external environment, an auxiliary drum storage system is provided. This system contains 3,244,032 bits of storage divided among eight cylinders each consisting of six fields of 33-bit words. Each field has 2,048 words distributed around the circumference of a cylinder. Five magnetic tape units of the IBM *728* design and a large cathode-ray tube display system are also provided. In order to obtain a reliability capable of providing continuous 24-hour operation, the whole machine, with the exception of some of the input-output equipment, is duplicated. This guards against over-all system catastrophies caused by sudden machine failures and allows planned partial shutdowns for maintenance purposes.

One major consideration in the design was the availablity of components, particularly in the case of the high-speed memory. The original thinking on this computer resulted in a need for a memory faster and larger than in any existing computers. This pointed toward the magnetic core memory development which had been under way for some time at the Massachusetts Institute of Technology (MIT). The logical design of the computer was centered around the engineering judgment that the minimum memory cycle time would turn out to be in the order of 6 microseconds and that the largest matrix of cores should be 64 by 64.

The availability of components also affected the choice of the high-capacity storage medium for which magnetic drums were chosen. The drum which seemed the most attractive and the one which was chosen was that used on the IBM *650*. The vacuum tubes and circuitry in the computer were developed from designs previously made by the Digital Computer Laboratory at MIT and the Electronic Data Processing Machine (EDPM) development laboratories at IBM. The redesign of these basic circuits was dictated by the increased reliability requirements. In many cases this necessitated the use of more vacuum tubes and other components than are required in scientific or commercial applications.

The features of the computer which are the primary subject of this paper are in the logical design areas. The majority of these features were dictated by the need to obtain the most efficient use of the drum capacity and core-memory speed with the minimum number of circuits. As a consequence, special attention was given to overlapping operations within the machine and to the balance between the available components and the logical arrangements which would maximize their usefulness.