DOCUMENT RESUME

ED 077 252 EM 011 186

AUTHOR Holm, Cheryl; And Others
TITLE CAISYS-8- A CAI Language Developed For A Minicomputer.
INSTITUTION Texas Univ., Galveston. Medical Branch.
PUB DATE 73
NOTE 26p.

EDRS PRICE MF-$0.65 HC-$3.29
DESCRIPTORS *Branching; *Computer Assisted Instruction; Computer Programs; Computers; Costs; Medical Education; Program Descriptions; *Programing Languages
IDENTIFIERS CAISYS-8; Compilers; Minicomputers; University of Texas Medical Branch

ABSTRACT

The University of Texas Medical Branch developed a minicomputer-based computer-assisted instruction (CAI) system which employed a teacher oriented software package called CAISYS-8, consisting of a highly modularized teaching compiler and operating system. CAISYS-8 used instructional quanta which generalized the flow of information to and from the student and encouraged feedback between student and teacher. Individual accumulator registers recorded student responses and directed program branching, and teachers were provided with a completely dichotomous tree structure. The system was found to be highly useful because the minicomputer itself was far less expensive than larger computers and less prone to obsolesence and because the compiler teaching language, at once simple, powerful, and flexible, was readily accepted by teachers who learned to use it in a short time. (PB)

# CAISYS-8 - A CAI LANGUAGE DEVELOPED FOR A MINICOMPUTER

CHERYL HOLM, WILLIAM M. THOMPSON, MARTIN C. WILBER* AND C. H. WELLS‡

Medical Education Computing Center
University of Texas Medical Branch
Galveston, Texas 77550



*Department of Orthopaedic Surgery, University of Texas Medical Branch, Galveston, Texas.

‡Department of Physiology, University of Texas Medical Branch, Galveston, Texas.

Abstract

A minicomputer based Computer Aided Instruction System is described. The system employs a new, teacher oriented software package called CAISYS-8. Programs written by teachers, instead of by programmers, are processed by the CAISYS-8 Compiler to yield computer automated teaching programs. The CAISYS-8 Operating System accepts and administers traditional and tutorial programs, simulation programs, drill type programs and automated testing programs. Key words: Minicomputer Compiler Teaching Language Computer Aided Instruction.

## INTRODUCTION

In the context of this paper, Computer Aided Instruction means the process of transmitting knowledge of a subject from an electronic computer system to a person previously ignorant of this knowledge. That is, the computer teaches the student a subject he knew little or nothing about before his encounter with the computer; and, after suitable programs are loaded into it, the computer accomplishes this task without human intervention.

Defined in this manner, Computer Aided Instruction (CAI) offers human culture a tool of such value and far

reaching implications that it staggers the imagination!

However, even partial realization of the potentials of CAI will require several decades of development and orientation in both the computer programming and the teaching fields.

When proffered the claims of the above paragraphs, a teacher unfamiliar with CAI usually responds with candid disbelief; and a programmer (even one who is familiar with existing CAI systems) responds much in the same manner. The former may be very effective at teaching students algebra, but hasn't the foggiest notion of how one speaks to computers. The latter may be proficient at speaking binary to a computer, but incapable of organizing algebra in such a manner that a student can learn it.

Programming professionals wait for teachers to design CAI curricula in a format which can be easily dispensed to students on computer terminals. Teachers wait for meaningful software packages which will permit teachers to teach via machines, meanwhile rejecting the hypothesis that their teaching methods can be emulated by a cold, impersonal and incomprehensible machine.

From a technical point of view, the required catalyst which will drive the teacher-programmer reaction to a useful product is a CAI COMPILER. Furthermore, such a compiler must allow a teacher to

directly design and implement his instructional materials entirely in the absence of programmer consultations or translations. In short, the compiler must provide a simple, yet powerful teaching language which can be managed entirely by the teacher.

Given, then, an analytical definition of "good teaching", a compiler or a set of compilers could be readily constructed and the entire process could be accurately defined.

Scholars and educators, of course, have argued this issue for several centuries, and an acceptable definition of "good teaching" has not been achieved. CAI systems designers must therefore be content with the slow and laborious process iteration. That is, using the most reasonable set of assumptions at hand, compilers can be constructed and tested. Results of tests are fed back to compiler designers, and the original compiler is modified to yield an improved version. Properly controlled, this iterative process converges to a better and better CAI system. But a very large amount of time may be consumed in the process. An extremely exciting aspect of this iterative process of compiler development is that perhaps now we are on the verge of discovering what "good teaching" really amounts to. When "good teaching" is achieved by computer control, thorough analysis can be freed from the subjective elements

which have stymied historical efforts at definition. The computer's intrinsic ability to monitor and record a student's responses while simultaneously tutoring him provides a source of objective data heretofore unobtainable.

However, it is well known that not all iterative processes converge. If the original assumptions are faulty, or if corrective feedback is not properly interpreted and utilized, then the CAI system will probably deteriorate as time passes.

The original assumptions leading to compiler design must include a wide variety of considerations.

Modularity is a critical aspect of compiler design. In any complex software system with intertwining and overlapping functions, even a seemingly minor modification often introduces ripples of incompatibility which can only be resolved through massive rewrite. Hence, applying feedback modifications to a non-modularized compiler is usually extremely difficult. Modularity of functions minimizes this problem by isolating troublesome areas. With modularity, old sections can be more easily altered, and new sections can be more readily inserted. Fortunately, CAI compilers require only a small fraction of the total machine time demanded by business or scientific problem solving compilers. Modularity of the CAI compiler, therefore, can and should be achieved even at the expense of compiler efficiency.

Another important consideration is the method and mechanics by which teachers submit their programs for compilation. If one considers, even momentarily, the thoughtful planning required of a teacher during program design, then the apparently inescapable conclusion is that a teacher should be able to write a program with pencil and paper. However, this conclusion did escape quite a number of early CAI compiler designers. Instead of pencil and paper design in the quiet of their offices or homes, teachers were often asked to sit in front of a teletype or CRT terminal and interactively build their programs while on-line to a computer. This process is not only exhorbitantly expensive from both a hardware and a software point of view, it is distracting and frustrating from a teacher's point of view. The compiler language must be a tool with which a teacher can become intimately familiar; and he must be able to use the language at any available moment.

Most critical, however, are two aspects of compiler development which are tied for first place in priority. These are cost of operating the system and acceptability of the compiler language from the teacher's viewpoint. CAI systems which have floundered in the past and some of today's systems which are surely not converging to successful systems failed primarily because one or both of these priorities received inadequate attention.

It is obvious that a compiler language for programmers instead of teachers will not achieve success. Teachers do not now and cannot in the future be expected to demonstrate genuine interest in CAI systems which require a third party (a programmer) as a translator between instructor and student. Such systems tend to strip the personality and creativity of the teacher from the instructional materials. In at least one respect, "good teaching" will never differ markedly from its present state: It must always be an individually creative task, and a teacher must be able to feel pride in his personal achievement. Submission of a set of disjointed notes to a programmer or filling in the blanks in a preformed teaching skeleton will neither engender pride nor promote a sense of personal accomplishment in teachers.

It should be equally obvious that even a good compiler language has little chance of success if it has to be implemented on a medium or large scale computer system whose cost is prohibitively high. Early development in electronic computing technology was directed primarily toward large computer mainframes and the enormously powerful central processors required for business and analytical problem solving tasks. Early CAI systems developers employed large computers because it was a forced option. Nothing else was available. In fact, much of the early effort to produce CAI systems

was provided by computer companies themselves and not by educational institutions. Few institutions could afford to develop and operate CAI software for the large and expensive computer systems.

For the last half-decade the electronic computing technology thrust has been toward the economical, yet versatile minicomputer. The minicomputer has not only provided CAI systems capability at a realistic price, it provides a hardware system which is vastly superior to large scale computers in many other respects. It is a better time-sharing and communications system than most large scale processors. With 12 or 16 bit words, the minicomputer is more efficient and usually faster than large scaled computers because serial strings of ASCII or EBCDIC are the fundamental language elements of CAI systems. CAI systems require logic processors, and perform quite well with minimal computational capability. Large scale scientific computers are designed primarily as arithmetic processors for 32 to 60 bit words. The minicomputer typically requires considerably less environmental control and can be employed as a portable CAI system. Hardware systems down-time is reduced to an absolute minimum with minicomputers, and usually amounts to only a few percent of the down-time experienced by large and complex computer systems.

# THE CAISYS-8 LANGUAGE

During the Spring of 1971, development of a minicomputer based CAI system was initiated on the campus of the University of Texas Medical Branch in Galveston, Texas. The project was a pilot study designed to test and evaluate the efficacy of minicomputer based teaching systems.

Initially, the hardware procured for the test system included a PDP 8/E Computer with eight thousand words of 12 bits per word memory, a console ASR 33 Teletype and a random access Kodak RA960 35mm slide projector capable of accessing a maximum of eighty 35mm slides. An interface was developed locally to place the projector under full computer control.

A Computer Aided Instruction System, CAISYS-8 was then developed over a period of approximately nine months. The software system was written entirely in the assembly language of the PDP 8/E. CAISYS-8 consists of a highly modularized two-pass compiler, an Operating System and a Loader.

The CAISYS-8 Compiler accepts input from a paper tape which is a direct ASCII copy of the pencil copy of a program submitted by a teacher. On the compiler's first pass, reasonably extensive diagnostic analysis of syntax errors is provided on an item by item basis, and a listing

```
01    BEGIN
02    1U
03    1S 3, 4, 1U2S
 :      :
 :      :
 :      :
10    2S 5, 6
11      "1",  1U3S,1A
12      "2",  1U4S
13      "3",  1U5S
14      "4,5",        1U7S,,0
15    P 1U8S
 :      :
 :      :
 :      :
20    53S 75
21      "SUPRACONDYLAR FRACTURE",     2U1S,1A,M
22      "CLOSED FRACTURE",            1U56S,,M
23      "FRACTURE",                   1U56S
24    P 1U54S, 1U55S
25    54S 76,R
26    55S 77
27      "FRACTURE",    1U76S
28    P 1U57S
29    56S 78
30      "SUPRACONDYLAR",    2U1S,1A
31      "SUP//COND/LAR",    2U1S,1B
32    P 1U57S
 :      :
 :      :
 :      :
40    2U
41    1S 333, E
 :      :
 :      :
 :      :
50    IF(A-5) E, 1U78S, 1U78S
51    IF(B-1) 2U3S, 1U78S, 1U78S
 :      :
 :      :
 :      :
99    END
```

Fig. 1. CAISYS-8 Compiler source program coding submitted by a teacher which describes the logical flow of a teaching program.

TXEL 3
PLEASE TURN TO PAGE 45 IN BINDER FOR X-RAY.

TXEL 4
PLEASE TURN TO PAGE 51 IN BINDER FOR X-RAY.

TXEL 5
PLEASE TURN TO PAGE 23 IN BINDER FOR
PHYSICIAN'S REFERRAL LETTER.

TXEL 6
NOW THAT YOU HAVE A BRIEF HISTORY OF THE PATIENT,
WHICH OF THE FOLLOWING DO YOU WISH TO PURSUE?
1. OBTAIN FURTHER HISTORY
2. PERFORM A PHYSICAL EXAM
3. OBTAIN DIAGNOSTIC STUDIES
4. SEEK CONSULTATION
5. INITIATE TREATMENT

TXEL 75
WHAT IS YOUR DIAGNOSIS? PLEASE BE BRIEF (I. E.,
SPRAINED ANKLE).

TXEL 76
SORRY, I DO NOT UNDERSTAND THAT RESPONSE. DID
YOU MISSPELL IT? IF NOT, REPHRASE IT. IN ANY
CASE, TRY AGAIN.

TXEL 77
THAT RESPONSE IS NOT CLEAR TO ME EITHER. WAS IT
A FRACTURE? DISLOCATION? WHAT?

TXEL 78
THE DIAGNOSIS OF FRACTURE IS CORRECT. WHAT TYPE
OF FRACTURE IS IT?

TXEL 333
SUPRACONDYLAR FRACTURE IS CORRECT!

Fig. 2. Text elements which are referenced by the CAISYS-8 Operating System during interaction with a student.

of the complete program is generated. If no errors were detected, then the compiler proceeds to the second pass. During the second pass, illegal or undefined references are flagged and, if both passes were error free, then binary output is generated.

The compiler's binary output consists of a series of logical blocks. Each block represents a quantum of instructional logic. For example, Line 3 of Fig. 1 is a quantum which presents a set of instructional materials to a student. The generalized format of this type of quantum is

$$N \quad S \quad t_1, t_2, \ldots t_i, \ldots t_f, \text{ vUnSm}$$

where N is an integer identifying this quantum; S indicates that text elements are to be presented to a student; $t_1$ is an integer naming the first text element to be presented, followed successively by text elements $t_2$ and so forth until, finally, text element $t_f$ is presented to the student. In general, $t_i$ is an integer which names the $i^{th}$ text element in a list of all text elements in the program. The text elements may be referenced in a completely random fashion. The commas between the text element names indicate that the CAISYS-8 Operating System will wait for a response from the student before presenting the next text element. The compound symbol vUnSm is a branching vector where each lower case letter assumes an integer value, and the vector points to text element m of instructional quantum n of Unit v. This entire

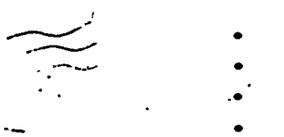
instructional quantum is called an "S-Statement". A collection of S-Statements may be conveniently grouped into a Unit and designated by U.

Thus, Line 3 of Fig. 1 will present to a student text elements 3 and 4 before branching to Line 10 of the program. The student studies each element at his own speed, then strikes a key to call for the next text element.

When the teacher desires to interrogate the student, another instructional quantum, the "S-Question", is used. For example, Line 10 of Fig. 1 is the first line of an S-Question. Note that several lines are required to complete this quantum. It has the general format

N S $t_1$, $t_2$, . . . $t_f$

"first anticipated answer", B, kA, X



"last anticipated answer", B, kA, X

P B1, B2 . . . BN

where the first line is defined exactly as in the S-Statement, but the branching vector is deleted after the final text element. The CAISYS-8 Compiler assumes that $t_1$, $t_2$, etc., are preliminary statements, and that $t_f$ is a question which the student is expected to answer. If the teacher desires, the preliminary text elements may be omitted as in Fig. 1, line 20, and the compiler will

assume that the remaining text element, $t_f$, in this instance 75, is the question. The deletion of the branching vector on the first line keys the compiler to interpret a multi-line S-Question.

In the second line (see Line 21 of Fig. 1) of the S-Question, the teacher enters, in quotations, any answer he predicts that the student might give in response to the question. It is important to point out that the "first anticipated answer" does not, in any sense, imply that this is either the most probable answer or the correct answer. Anticipated, in this context, means only that this answer, whether right or wrong, is predictable on the basis of the teacher's experience with this particular question.

Following the answer in quotations, B indicates a general branching vector as defined for the S-Statement. If the student's answer matches this anticipated answer given by the teacher, then this branching vector determines where the next point of instruction is to begin.

Next, after a comma, k is an integer talley, and A is any of several accumulators which the teacher may wish to use. If the student's answer matches this anticipated answer given by the teacher, then the accumulator designated by A will be incremented by the value of k.

Last, and again set off with a comma, X is a single

letter code which directs the CAISYS-8 Operating System to use a particular method of matching the student's response with this anticipated answer entered by the teacher. If X is omitted, then the student's response must exactly (spaces are ignored) match the teacher's anticipated answer. If X is M, then the student's answer must consist of exactly the same words used in the anticipated answer, but the student's words may be given in any order. If the teacher uses several words in the anticipated answer, and X is O then a single word response by the student will match if the response word matches any word in the anticipated answer.

The teacher may use many different anticipated answers, each having an individual accumulator and each being evaluated by an individual matching scheme. Furthermore, a teacher may substitute slashes (/) for any letters in the anticipated answer to force a match with a student's response. That is, if the teacher employs "conc//ve" as an anticipated answer, then a student response of either concieve or conceive will match the teacher's answer.

As pointed out earlier, the choice of sequence of anticipated answers provided by the teacher is not in any way restricted. The answers are, however, analyzed in sequence from top to bottom by CAISYS-8. It is

therefore possible to design comprehensive answer sets by carefully organizing the answers in sequence and by varying the matching algorithms employed.

A variety of other student response analysis algorithms are being evaluated. When a new algorithm proves useful, it can be incorporated into the modular structure of CAISYS-8 without difficulty.

Following the set of predictable answers, on the next line (see Line 24 of Fig. 1), P defines a prompting sequence which will be employed if the student's response does not match any of the answers given by the teacher. B1, B2,....BN is a set of general branching vectors and they are used in ascending numerical order. If the student's first answer cannot be matched, branching vector B1 points to some S-Statement that will "prompt" the student in some fashion, and then the original question may be again asked. If the student's second response cannot be matched, then B2 points to another S-Statement which will again prompt the student. This process continues until, either the student's response matches one of the anticipated answers or finally, BN is employed, for example, to tell the student the correct answer or move to a remedial section of the program for further instruction.

A teacher writes a CAISYS-8 program by stating "BEGIN", and, on the next line, giving a Unit number.

S-Statements and S-Questions may be grouped in Units of any size. The entire program may be a single Unit, or each question may be interpreted as a distinct Unit. To terminate a program, the teacher simply writes "END".

To enable a teacher to discern trends in a student's responses, the accumulators may be evaluated at the end of each Unit. A statement was borrowed from FORTRAN to provide this feature. When the teacher desires to evaluate an accumulator, an "IF Statement" is employed. This statement tests the value of an accumulator and branches conditionally. This statement has the general form

IF (AC - n) B1,B2,B3

where AC is any accumulator, and n is any integer test value. (In CAISYS-8 AC is specified by a letter of the alphabet, i.e., A, B, C, etc. designate distinct accumulators). The CAISYS-8 Operating System performs the subtraction indicated in parentheses and then uses the general branching vector B1 if the difference is negative, B2 if the difference is zero, and B3 if the difference is positive. In Line 51 of Fig. 1, for example, the program will branch to 1U78S if accumulator B has tallied a count of 1 and this statement is executed.

In addition to the general branching vector, several special vectors may be used in certain instances. If "E" is used as a branching vector for an S-Statement, then

the CAISYS-8 Operating System moves to the first IF Statement at the end of the current Unit. Note that several IF Statements may be used, one following another, at the end of a Unit. If an E vector is used in an IF Statement, then the operating system moves to the next IF Statement (see Line 50 of Fig. 1). It should be emphasized that the accumulator system is invisible to the student. Only the teacher is aware that conditional branching and scoring is being employed in the instructional materials.

An "R" may be used as a vector to terminate an S-Statement. This use of the R vector (see Line 25 of Fig. 1) converts an S-Statement into something of a subroutine; because it indicates that, upon completion of presentation of text elements in the S-Statement, the program must immediately return to the statement which "called" or branched to the S-Statement. This vector affords great economy when employed as a branching vector to select statements that are frequently used to prompt a student. The prompting statements may be written only once, but employed at any or all questions in the entire program. Computer memory is conserved, and a teacher is required to write trivial prompting comments only once.

A feature provided for the student is review of previously viewed text elements. At the teacher's option,

a student may enter a request to view a predetermined number of the text elements viewed earlier.

As another feature, the teacher may wish to inform a student that some previously viewed text element may be reexamined, at the student's option.

Another extremely powerful and attractive feature of the CAISYS-8 Operating System is the calculator package which a student may summon and employ to solve analytical problems presented to him. FOCAL, a beautifully simple yet surprisingly capable software package produced by Digital Equipment Corporation, resides on a mass storage device and can be called by a simple command. When FOCAL is called, the entire CAISYS-8 Operating System is written out to mass storage and FOCAL is read in. When the student has completed his numerical work in FOCAL, he calls the CAISYS-8 Operating System back into core and resumes the program at his earlier departure point. Hence, the teacher may ask for extremely sophisticated calculations on the part of the student; and the student needs only to call FOCAL, compute the required answers, and return to CAISYS-8 to answer the questions.

## DISCUSSION

The minimal hardware configuration described previously was placed in operation in January of 1972 and has been used regularly by students for more than a year. In this minimal configuration, with only eight thousand words of storage available, all text elements are presented on 35mm slides, and lacking mass storage, FOCAL is not available. An expanded PDP 8/I Computer System has been used to continue software development. This larger system includes 16K of core, two DECTAPES, a DECDISK and a Tektronix 4010 Graphics Display Terminal. A slide projector is not included.

A text element in the larger system can be either printed text on a teletype or printed text and graphical displays on the Tektronix terminal. To program for the smaller PDP 8/E system, a teacher submits a program of the form shown in Fig. 1, and text element numbers are assumed to mean 35mm slide numbers. To program for the larger PDP 8/I system, a teacher must, in addition, submit a list of text elements individually identified as TXEL 1, TXEL 2, etc. This list is stored on a mass storage device and the text element numbers given in the program are assumed to reference this list of text elements. An example of such a list is given in Fig. 2.

The present CAISYS-8 system supports one student terminal and it is not a time-sharing system. This is

because the hardware available for this pilot study was absolutely minimal. However, CAISYS-8 was written with a view toward multi-terminal time-shared usage. The logical blocks of instructional quanta were designed to be completely self-contained; each block contains complete specifications for all terminal traffic handling. The CAISYS-8 Operating System was designed as a re-entrant system for processing instructional blocks randomly selected by individual student terminals.

Similarly, the limited storage available precluded the possibility of an accounting package, log-on and log-off routines, permanent storage of all student responses, and many of the trace routines which are essential to any good CAI software system.

Acknowledging these and other shortcomings, after two years of evaluation of the system, it is felt that CAISYS-8 superbly demonstrates the feasibility of teaching compilers for minicomputer based Computer Aided Instruction.

Two distinct types of CAISYS-8 programs have been presented to medical students. One program type is basic medical science teaching, using 35mm slides to present text elements.[1] The other type is patient simulation. Patient simulation has intrigued a number of CAI enthusiasts for several years. The primary impetus for efforts to achieve computer simulation of sick patients seems to be of great value in training young physicians

## CONCLUSIONS

Teacher acceptance and dexterity with CAISYS-8 have convincingly demonstrated that teachers will provide the teaching logic and course programming if they are given a suitable compiler language.

These very significant facts have emerged as a result of the development of CAISYS-8:

First, powerful and versatile Computer Aided Instruction compiler languages, designed for teachers, can be developed over reasonably short periods of time. The software costs involved in such a development are considerably less than any presently available alternative approach to CAI system implementation. Instead of investing dollars for programmers who operate and maintain exhorbitantly expensive commercial systems, it is far wiser to invest in programmers who will develop and maintain a system which is owned by the user.

Second, purchasing a minicomputer hardware system whose useful life exceeds fifteen years is a sound practice. The total cost of a minicomputer system with eight terminals is typically less than $60,000. This cost is equivalent to only several months rental paid on medium or large scale computers (usually business machines) which, as a rule, are obsolete in five years! Unquestionably, the minicomputer is a milestone of first

in the decision making related to both diagnostic and therapeutic medicine. Several CAISYS-8 programs have been devised which provide this type of training. Student response has been very encouraging and usage of these patient simulation programs steadily increases in spite of the fact that only the pilot project hardware is available. This type of programming has surprisingly proven to be a very important test of the CAISYS-3 language. The language was not designed with any thought of providing patient simulation. In spite of this, a surgeon with no prior knowledge of data processing techniques acquired genuine competence with the language after a few hours of training and a few days of practice. He has subsequently produced several patient simulation programs using 35mm slides for text, general and special physicals, laboratory and X-ray examinations, laboratory reports, X-rays and treatment parameters.

magnitude importance to the field of Computer Aided Instruction. This means, simply, that CAI systems need not, any longer, be a luxurious burden on educational institutions. Implementing CAI on a minicomputer system can place this powerful educational technique in schools at a cost which is competitive with standard teaching methods.

The enormous instructional advantages and improvements offered by minicomputer based Computer Aided Instruction will, within the next several decades, become so forceful that few educational areas will be able to resist implementation of this method of instruction. In elementary education, for example, a veritable revolution of instructional techniques is now within view; and these schools will unquestionably evolve to instructi n centers whose CAI techniques will practically obliterate the traditional methodology.

## REFERENCES

1. C. H. Wells, Teaching renal problem solving on a minicomputer based instructional system, J. Clin. Comput. (in press).

## SUMMARY

The CAISYS-8 Computer Aided Instruction System consists of a highly modularized teaching compiler and operating system. By means of this software system, a teacher can prepare teaching materials and program a minicomputer based hardware system to teach students.

The compiler teaching language is simple, yet extremely powerful and flexible. A teacher can, in a few days, become quite adept at writing fairly sophisticated teaching programs. CAISYS-8 employs instructional quanta which were designed to generalize the flow of information to and from the student, but the rigid historical CAI concepts of "right" and "wrong" answers was avoided entirely. Instead, the CAISYS-8 teaching compiler encourages a "give and take" relationship between teacher and student. In addition, individual accumulator registers are provided for recording each predictable response from a student. These registers can be easily employed by the teacher to direct future branching within the program.

From the teacher's point of view, the CAISYS-8 language provides a completely general dichotomous tree structure. Within this structure, a teacher can implement a program as simple as a multiple choice test, or as sophisticated as a program to teach elements of mathematical field theory.

The CAISYS-8 system was developed because the thrust of electronic computing technology toward the minicomputer has, at last, provided a hardware system which can be economically justified for teaching. With these small, but versatile processors, Computer Aided Instruction can be implemented on purchased, dedicated machines at a one-time cost equivalent to only several months rental paid on the earlier medium or large scale computers. Furthermore, the purchased minicomputer now offers CAI systems developers reasonable immunity to the vagaries of processor obsolescence which has so long beset the medium and large scale computer field.