AN INTERACTIVE PSEUDO-ASSEMBLER FOR INTRODUCTORY COMPUTER SCIENCE

Ted Sjoerdsma
Department of Computer Science
The University of Iowa
Iowa City, Iowa 52242

INTRODUCTION

Since 1968 the University of Iowa Computer Science Department has used a locally developed Easy Assembler SYstem (EASY) to accomplish a better comprehension of the concepts taught in the segment of the introductory computing course which dealt with internal structure and organization.

EASY served its purpose to a widely varying degree. The degree of success depended primarily upon the instructor's acceptance of EASY and a careful understanding of its purpose for the course. Two main problems arose in the continued use of EASY: 1) the instructor (TA or faculty member) had to become very familiar with a narrowly used language and pseudo-machine; and 2) the amount of time spent on EASY as a language often far outweighed the benefits relative to other necessary segments of the course. Thus EASY became an object of contention among faculty and teaching assistants and consequently, among the students in the course.

In the fall of 1973 my attention focused on EASY (with all of its problems) as a likely candidate for a computerized-interactive-tutorial segment of this course. Since the consistency of use and precision in presentation were important aspects of teaching the concepts related to EASY, such an approach seemed natural.

TUTORIAL MATERIALS

The vehicle used for the tutorial interaction with the student is the Course Writing Facility (CWF) [1] on the Hewlett-Packard 2000F computer. This facility is linked with the Instructional Management Facility on the same machine for purposes of course management and statistic gathering.

The actual implementation of the tutorial uses a variety of approaches interwoven into a short course, called TeachEASY. These different approaches include a prerequisite test, a student handbook, review at student request, assignment of student programming, step-by-step analysis of one program, and acceptance of student programs for the EASY system.

Program Description

The tutorial solicits student input and insists on a correct answer before proceeding. The answers are in the form of multiple choice, fill in the blanks or reply to a question. (See example, Figure 1). Anticipated wrong answers trigger a hint and unanticipated answers usually supply the correct answer. Partial answer processing capabilities allow realistic interaction. (For example, when two words are required for the object code in an answer, if one word is correct the student can be told this fact and asked to change the other word.)

The student is able to sign-off at any time that the program is expecting input, as well as at break points. The re-entry point is always at the beginning of the last major segment unless the student signed off at a break point. In the latter case, the re-entry is at the beginning of the next segment. (See flowchart, Figure 2.)

After receiving each of the programming assignments the student may choose to review any portion of the course. Once the student chooses to review, he or she remains in the review mode

Figure 1 Samples of Interactions between Student and TeachEASY

1.

```
 OK. Here's an instruction : START    LOAD  5, HERE
There is an error. Type the letter of the field you think
it is in. (a)Label (b)Operation (c)Operand(s)
a
No . The label field is OK. Try again.
c
Yes . The operand(s) field has a forbidden blank in it.
```

2.

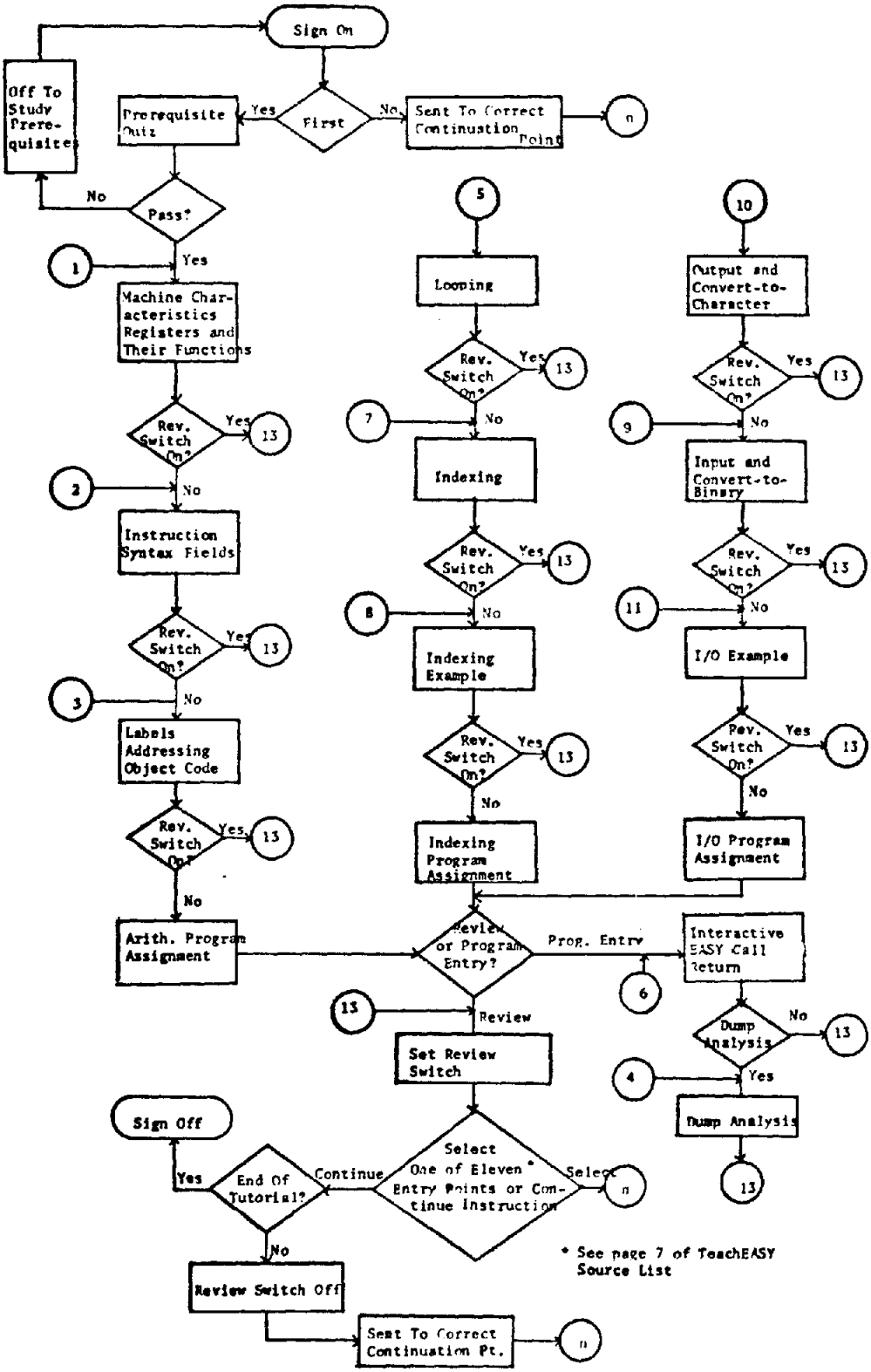
```
 Now we have gone through a first pass that an assembler
usually makes i.e., assigning addresses and in particular,
assigning addresses to labels. Let's make a label table.

        LABEL          LOCATION
        A                000
        TEMP             00B     (notice: B, not 8)
        TEN              015
        ONE              016
        ZERO             017

Check this table (and copy it if you are on a CRT).

Type 'Ok' when ready to proceed
?ok
 Now using the label table and the operation codes
found on page 5 , we can produce the object code.
For the instruction  A  LOAD  5,ZERO   what is
the operation code for LOAD ?
10
No. Try again.
1E
Right.
What register is used in  A  LOAD  5,ZERO  ?
5
 Sure.
 Now we notice that no index register is used,
so we indicate this by supplying a zero, and we have
produced the first word of the object code. It is 1E50.
The second is made up of the address of ZERO.  Look it
up in the table we produced and give it to me.
0017
Right.
 The object code of  A  LOAD 5,ZERO is 1E50 0017 .
Now see if you can give me the object code for
   ADD  5,ONE
1E50 0016
You have the address correct, but something is wrong
in the first word . Try the entire code again.
1050 0016
 Excellent.
Would you like to review any section of this course?
Type yes or no.
```

Figure 2 TeachEASY Flowchart



until the option to continue the course is selected. This allows the student to continue review of several segments.

When the student desires to enter a program, the Course Writing Facility calls in the Interactive EASY System by a simple function call to *EAZ1. After the program entry, assembly, editing, execution, and desired reruns, the control is turned back to CWF. At this point a dump analysis segment may be chosen.

The tutorial makes study assignments in the Interactive EASY Handbook [2] and frequently references programs and data in it. This requires the handbook at the terminal.

The tutorial concentrates on the concepts relating to Computer Organization: general machine structure; instructions, their codes, and the action they produce; operations; arithmetic; branching; indexing; input and output; and data representation.

Revisions

After the first student runs were completed, it became obvious that the last few segments were much too wordy and required too little student interaction. Students indicated this by repeating the segments several times, or by telling me that the segment was too much like a lecture. Some immediate changes were made to the text which required student responses, thus becoming much more palatable for later student runs. However, more of this type of change still must be made to the sections on indexing and input.

Writing Time

Initial writing time of the interactive programs, which produced an estimated 3 hours of tutorial materials, was approximately 160 hours. The revisions made after test runs and after initial student runs required approximately 30 hours of effort. There will still be several hours needed to continue refining the materials into finished product form.

INTERACTIVE EASY

The Interactive EASY System is made up of six programs written in H-P Time-Shared BASIC which are "chained" together and are invoked by the student through the interactive tutorial materials in CWF by means of a function call. These six programs [2] are the implementation of a system designed to accomplish the following:

1. Accept and syntactically analyze each student program statement as entered; reject with an error message if incorrect.
2. Internally build a label table and statement table which can be displayed to the student on request.
3. When the final program statement (END) is entered, allow the student to edit (i.e., delete, replace or insert) his statements, which, in turn, causes the system to make necessary changes to label and statement tables.
4. Assemble the program into machine code, statement by statement (on display), stopping at unresolved references.
5. Execute any correctly assembled program.
6. Produce an error message and a dump of registers and memory if error occurs in execution. Provide a dump upon request for any execution.
7. Allow the student to re-edit, re-assemble, and re-execute.

Interactive EASY uses a limited subset (see TABLE 1) of the operations found in the batch form. (This subset was determined through classroom experience over several years and is identical to the set of operations, with 1 or 2 obvious exceptions, used in the batch mode.) This subset includes only integer arithmetic operations, a small set of compare and branch operations, binary and character conversion operations, and input/output instructions.

The file organization which allowed each student to have his or her own files was based on the port number assigned by the H-P system as the terminal accessed the system. There are four

TABLE 1. Interactive EASY Operations.

Operation	Operation	
LOAD	CMPRS	(Compare Register to Storage)
STORE	BU	(Branch Unconditional)
ADD	BE	(Branch if Equal)
SUB	BL	(Branch if Low)
MULT	BH	(Branch if High)
DIV	CVTB	(Convert to Binary)
INPUT	CVTC	(Convert to Character)
OUTPUT	HALT	

Assembler Operations

DC	(Define Constant)
DS	(Define Storage)
END	

files used by all students which were segmented according to the number of terminals available to the class. Thus even though 100-200 students would use the files, the number of file segments would still be limited to, say, 16 or 32. The files were required when building the student's unique statement table, label table and assembled program. The fourth file was used in updating during the editing phase.

EVALUATION

Pretest and Quiz

Prior to the experimental period (during which 20 of 92 students were selected to use the interactive tutorial system while the remainder use the batch form of EASY) a quiz was given over previous unrelated material. Also prior to the use of EASY a pretest was given which indicated almost a universal lack of knowledge concerning the concepts to be covered in this course segment.

Examinations

At the end of the experimental period all students took an examination over the materials and concepts covered in the first six weeks of the course (referred to as the "EASY exam"). All questions, except the first, were precisely related to EASY concepts taught during the experimental period.

The results of this exam are presented for each group in the following graph and in TABLE 2.

Figure 3. EASY Exam Results

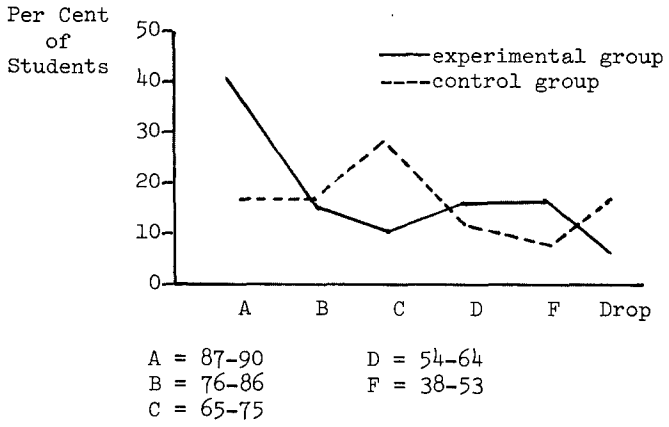


TABLE 2. Results of EASY Exam

Experimental Group			Control Group		
Number	Grade	%	Number	Grade	%
8	A	40	12	A	17
3	B	15	17	B	17
2	C	10	20	C	28
3	D	15	8	D	11
3	F	15	5	F	7
1	drop	5	10	drop	14
20			72		

Note: 3 students dropped the course prior to the experiment.

The results seem to indicate that the students in the experimental group performed as well or better than those in the control group. Admittedly the sample is much too small to draw valid conclusions.

The course final exam did not include questions about the concepts, per se, which were taught in the experimental period. However, the exam was comprehensive, and the final results should again indicate the relative abilities of each group. Both the quiz prior to the experiment and the course final examination indicate that the two groups were homogeneous and unbiased relative to ability.

Relative Time Considerations

At the beginning of the experiment the students in each group were provided log sheets to determine the amount of time involved in this segment of the course. The batch EASY group was

required to hand in a program log sheet for each of the assigned programs that they handed in. The interactive EASY group filled out a log sheet for each session at the terminal.

One of the items which could be checked for accuracy was the indicated amount of terminal time for each student using the interactive mode. Using the student log sheets, the total average terminal time per student was reported as 364 minutes (6 hours, 4 minutes). The actual average total terminal time as recorded by the Instructional Management Facility was 359 minutes. This five minute discrepancy indicates the accuracy of the student reporting.

Several time-related statistics gathered from the log sheet data are the following:

1. Program keypunch time (41.8 minutes average) was not much different from program entry time (41.0 minutes average).
2. Average program writing time for the interactive group was only about two-thirds of that required by the batch group (65.4 minutes average vs. 94.6 minutes average).
3. The most significant time difference is the total time involved. The average student time spent in the batch mode was 8 hours and 56 minutes. This compares with 7 hours and 9 minutes for the interactive mode, a full hour and 47 minutes less (see TABLE 3).

These time comparisons do not include unclocked study time, exam time, or time spent waiting for turnaround or available terminals.

TABLE 3. Time Comparisons

Item	Experimental	Control Group
Lecture		400 min. (8 × 50 min.)
Terminal	364 min.*	
Program writing	65.5 min.	94.6 min.
Keypunch		41.8 min.
Total	429.5 min. (7 hrs., 9 min.)	536.4 min. (8 hrs., 56 min.)

*Includes program entry time of 41.0 minutes.

It should be noted that the interactive EASY users had several adverse conditions in the experiment. These included bugs in the pseudo-assembler, and the tutorial materials. The most serious condition was a file separation problem which made it necessary for students to enter programs serially until the third week. This represented a serious time loss to some of the students.

Subjective Student Reactions

One of the less tangible items in an evaluation of a learning situation is the response of the students. An experienced teacher senses rather quickly whether or not the teaching is reaching the level of the student and leading him or her foreard. An interactive tutorial, however, does not have this type of human gauge--a teacher's feeling.

Historically, student reaction to the lecture-batch EASY teaching has been rather negative and the control group's reaction was still rather negative. On the other hand, seven students among the remaining nineteen in the experimental group volunteered a statement of their enjoyment of the tutorial on interactive EASY and its related concepts. Two students in the experimental group had taken the course before and had done poorly. One of these obtained an A in the EASY exam and the other (who obtained a B) said that if we had used this form of instruction the previous time, he would have done much better and not have had to repeat the course.

General solicited reactions from the experimental group were much more positive than those from the control group. This coupled with test scores indicates that the experimental method using CAI with an interactive pseudo-assembler was a more effective teaching mode.

SUMMARY, CONCLUSIONS AND IMPLICATIONS

Summary

An interactive pseudo-assembler was designed and implemented. An interactive tutorial course segment was written and interfaced with the interactive pseudo-assembler. Twenty students out of a class of ninety-five were selected to use this method of instruction while the remainder stayed

in a lecture mode using a batch pseudo-assembler. The students in each group used the same set of instructions, were taught and examined on the same concepts and were given the same programming assignments.

The information gathered before and after the experiment allowed comparison of student performance on an early quiz and on the final examination which demonstrated very similar results for both the experimental group and those in the remainder of the class. However, the results of the examination on the concepts taught in the experimental period showed that the experimental group did as well as, or better than, the control group.

Comparisons made with recorded time spent within each group showed a marked decrease in time spent by the experimental group. A study of time at the terminal versus the exam score obtained by the students in the experimental group showed little overall correlation but did indicate the students with the better scores spent less time at the terminals.

Conclusions

This study has established the following conclusions:

1. It was feasible to design and implement an interactive pseudo-assembler.
2. A computerized tutorial course segment was developed which, linked to the interactive pseudo-assembler, could take the place of a traditional course segment in the lecture-batch mode.
3. Students using the tutorial and interactive EASY learned the desired concepts as well, or better, than those students in the traditional mode.
4. The students who received the traditional lecture-batch instruction required considerably more time than those using the tutorial.
5. Within the experimental group using the tutorial, those students receiving the higher scores on the exam did not require more time at the terminal, but considerably less.
6. The cathode ray tube is the preferable terminal, with hardcopy terminals required only for producing program results that must be submitted to the instructor.
7. A large investment of faculty and programmer time is needed in a carefully prepared tutorial. The tutorial preparation required almost 200 faculty hours for approximately 6 hours of student terminal time. The production of Interactive EASY required 320 programming hours.

Implications for Computer Science Instruction

When one reviews the large number of topics which are recommended for inclusion in the computer science introductory course [3], it is immediately evident that these can hardly be covered well in a single semester course. This is especially true when more emphasis is being placed on the style and structure of programming.

Computer assisted instruction may well be the vehicle which will allow computer science teachers to offer a more complete course. The segment of the course developed in this study has already demonstrated the ability of CAI to provide equivalent or better instruction in less time.

The computer organization and data representation segment discussed in this study will continue to be used on a wider scale with future students. It, no doubt, will be revised and improved and, in some instances, might be used in parallel with the lecture sessions.

The prospects of using this method successfully on other course segments are bright. One can expect simple tutorials to be developed which could be used to teach such topics as the syntax of a language; data representations such as floating point numbers, vectors, arrays, etc.; basic programming concepts such as constants, variables, functions, expressions, etc.; and many other conceptual and factual topics.

As tutorial segments are prepared, careful consideration of essentials for inclusion becomes a natural process and thus extraneous topics are eliminated. This is forced upon the author by the large amount of time required for preparation of a good tutorial unit. This implies that the chaff

is separated from the wheat and the student is presented only essential topics.

Good materials prepared by master teachers can be improved to a high level of excellence and then be consistently available to all students. Initial time spent on developing such materials will result in better and more precise instruction with a payoff in time conservation and better instruction for the student.

NOTES AND REFERENCES

1. Hewlett-Packard Company, Course Writing Facility (HP24383A), Cupertino, Calif.: Hewlett-Packard Company, 1974.

2. Copies of TeachEASY, its documentation and the Interactive EASY Handbook are available from the author.

3. ACM Curriculum Committee, Curriculum '68, Comm. ACM 11(1968), 391-401.

Other Related References

Amarel, Saul, Computer Science: A Conceptual Framework for Curriculum Planning, Comm. ACM 13(1971), 391-401.

Austing, R. H. and Engel, G. L., A Computer Science Course Program for Small Colleges, Comm. ACM 16(1973), 139-147.

Blount, S. E. and Fein, L., The Practical Aspect of Computer Science Education--Discussion, Comm. ACM 16(1973), 45-46.

Brillinger, P. C. and Cowan, D. D., A complete package for introducing computer science, Proc. SIGSCE Symposium in Acad. Educ. in Computer Science, 1970, 118-126.

Feurzeig, W., et al., SIMON: A Simple Instruction Monitor, Washington, D.C.: Office of Naval Research, 1970.

Schurdak, J., An approach to the use of computers in the instructional process and an evaluation, Amer. Educ. Research J. 67(1967), 59-73.

Walker, M., A University-Level, First Course in Computer Science: A Case Study, Proc. of IFIP World Conference 1970, II, 389-396.

Weiner, L. H., Machine generation of assingments for a mass education introductory programming course, SIGSCE Bull. 5(1973), 181-185.