Effectiveness of Computer-Aided Learning as a Direct Replacement for Lecturing in Degree-Level Electronics

J. Nicholas Coleman, David J. Kinniment, *Member, IEEE*, Francis P. Burns, Timothy J. Butler, and Albert M. Koelmans

***Abstract*— The application of computer-aided learning as a direct replacement for, as opposed to an adjunct to lecturing, is still in its infancy in higher education. This paper examines some of the reasons for its slow uptake and then describes a project to develop courseware for a large proportion of the Electronic Engineering syllabus within several United Kingdom higher education institutions. The first modules to be completed cover the area of Computer Engineering. We describe the philosophy and design of this courseware, and then report a series of tests in which the examination performance of students using it was compared with that of control groups taught in traditional lectures. The results clearly suggest that carefully designed courseware can lead to a large reduction in teaching time, with no significant difference in learning. We then describe the development of this courseware into what is believed to be the first degree-level Electrical Engineering course module to be replaced in its entirety by computer-based self-teaching. We discuss the impact of this development on the course structure, and show how the time gained has been used for additional practical work and tutorial support. Animated excerpts from this material are available by anonymous FTP.**

***Index Terms*— Computer-aided learning, course administration.**

I. Introduction

In recent years a great many computer-based packages have entered the higher education curriculum. Many are intended to augment a traditional lecture-based course in order better to illustrate the principal theme of the syllabus, a range of examples of relevance to the Electrical Engineering disciplines being given in [1]. Particular examples of interest in microprocessor teaching may be found in register level simulators [2], and higher level simulators [3]. Other packages are directed at automated assessment, e.g. [4]. However, apart from a few current experimental developments, e.g. [5] for distance learning, these materials are not intended as a direct and entire replacement for the lecturing.

Observers have drawn attention to the fact that the uptake of these computer-based packages, even only as an adjunct to an existing lecture course, is somewhat restricted in higher education [6], [7]. As regards the more advanced activity in which computer-based "courseware" replaces university lecturing completely, successful examples are rare [8]. Reasons postulated include the lack of suitable courseware, lack of time and staff to develop new material, and lack of financial and administrative support to do so. Although these problems are obviously soluble, a number of more fundamental doubts about the desirability of this type of teaching have tended to deter university staff from making the necessary commitment to it.

Let us examine some of the objections to the replacement of lecturing by computerized self-teaching. The first concerns its effectiveness as a teaching method. Do the students learn as well from it as from traditional lectures? There is as yet little objective evidence on this point. Related to this is the fact that students in higher education are much less tightly supervised than their high-school counterparts, and a greater degree of self-discipline is therefore required of them if self-teaching is to be a success. Is it reasonable to expect sufficient commitment from them to make the experiment work? Another issue derives from the high degree of specialization of university-level courses, and the tight integration of the several courses comprising a particular degree. This tends to prevent the easy transfer of courseware from one institution to another, and thereby exacerbates the shortage of suitable material.

Against these objections, compelling reasons have been suggested for experimenting with the development of a fully computer-based course. A natural consequence of such a development would be a reduction in staff lecturing time, but there may also be a possibility of a reduction in the students' learning time. If the new teaching technique were also as effective as the lecturing that it replaced, then the time saved by both staff and students could be redeployed for individual tutorial contact. Under this hypothesis, the time required of the lecturer would not change, but the nature of his contact with the students would shift away from class teaching and toward individuals or small groups. The total learning time

Manuscript received November 30, 1994; revised March 13, 1998. Demonstrations of the computer-aided-learning material to which this paper relates can be found in the file ieeecal.zip, obtainable by anonymous FTP (binary) from ukhserv1.ncl.ac.uk.

J. N. Coleman and D. J. Kinniment are with the Department of Electrical and Electronic Engineering, The University of Newcastle upon Tyne, Newcastle upon Tyne, NE1 7RU, U.K.

F. P. Burns was with the Department Electrical and Electronic Engineering and is now with the Department of Computing Science, The University of Newcastle upon Tyne, Newcastle upon Tyne, NE1 7RU, U.K.

T. J. Butler was with the Department of Medical Statistics and is now with the Faculty of Medicine, The University of Newcastle upon Tyne, Newcastle upon Tyne, NE1 7RU, U.K.

A. M. Koelmans is with the Department of Computing Science, The University of Newcastle upon Tyne, Newcastle upon Tyne, NE1 7RU, U.K.

Publisher Item Identifier S 0018-9359(98)05816-6.

required by the students may also remain unchanged, and although they may find themselves in less overall contact with the lecturer, they would have new opportunities for attention which could be customized to their own needs. In other words, the lecturer would become less involved with the delivery of basic knowledge, and more with the facilitation and guidance of learning.

In 1992, starting from a base in which almost no suitable computer-based material was available, a collaborative project was established between several United Kingdom Electrical Engineering and Computer Science Departments with the aim of developing courseware for a large proportion of the identifiably common parts of the degree syllabus. In doing so, we had an opportunity to investigate the questions posed above. Standing midway between the creative arts and natural sciences, engineering was a suitable discipline for this experiment because of its reliance on large quantities of diagrammatic information. Although it was not felt that computer-aided learning could ever portray the creative design concepts pervading any advanced engineering activity, it could possibly express many of the fundamental techniques, ideas, and principles characterising the early stages of a degree education.

By 1994 the first piece of courseware was ready for use. It was entitled "Computer Architecture," and mounted on PC/Windows systems. During the Spring term, 1994, it was tested by dividing a class of Electrical Engineering students: half doing the courseware and half coming to the equivalent lectures. At the end of the trial, a short test was given to the entire class to see if any difference could be observed in the scores obtained by the two halves. A more detailed analysis was then made, which took into account their overall average for the year. Encouragingly, we found no statistically significant difference between the performance of the two halves, either at the end of the trial or at the end of the year. There was thus no aspect in which the CAL could be said to be worse than the lecturing. The remainder of this paper describes the courseware itself, the experimental procedure, and the results we obtained. We then describe the integration of the Computer Architecture module, together with other new material, into a self-taught Computer Engineering course, first used during 1996–1997.

II. The Courseware

Within the engineering disciplines, a degree course typically includes a series of specialized lecture modules of approximately one term's duration. The modules exhibit a complex pattern of interaction: some progressing sequentially, others reinforcing concurrent modules, and others having an all-pervading influence throughout the course. It is often a matter of some difficulty for the lecturing staff of the department, as they struggle to keep abreast of a rapidly changing subject, to keep the modules within a coherent framework. The resulting courses vary quite markedly between institutions, reflecting the different perceptions of priorities at each.

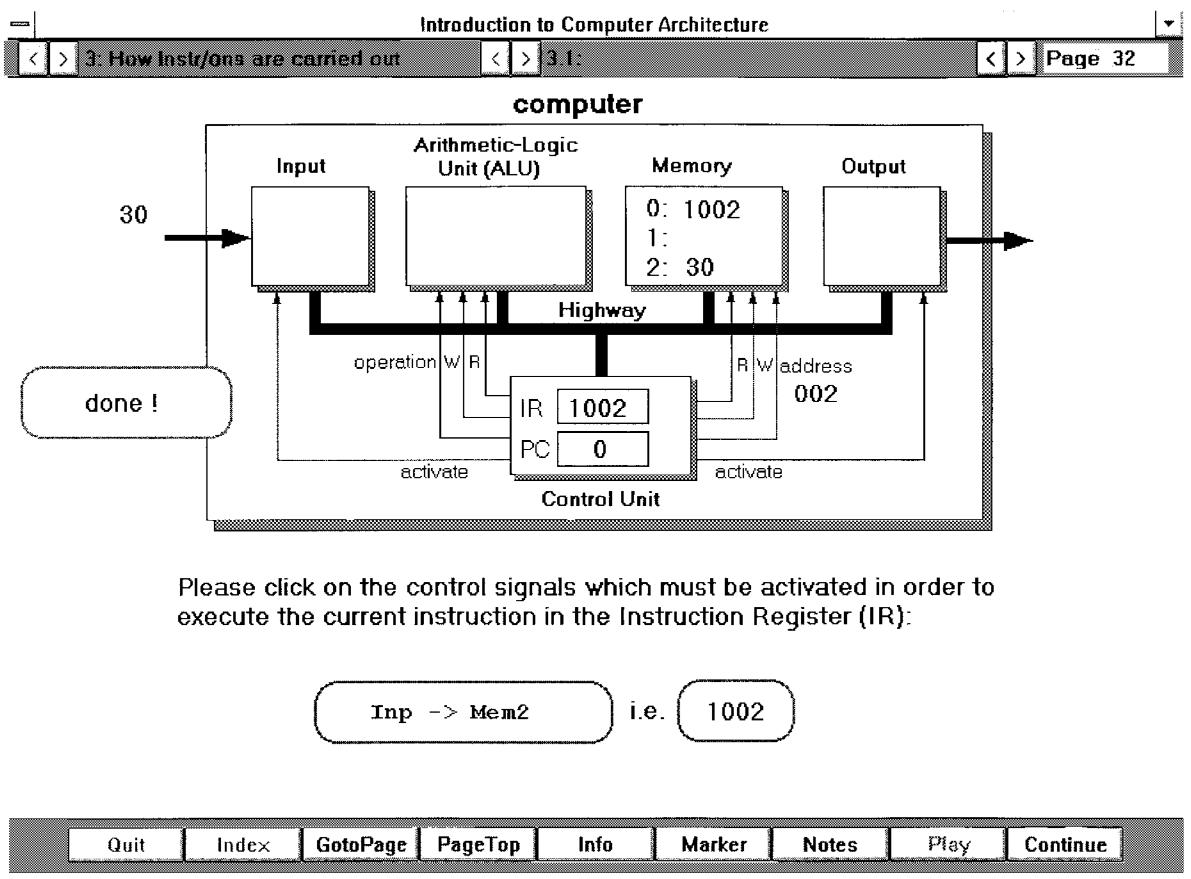
From the outset then, it was unlikely that a single suite of courseware could be appropriate for all degree courses in the subject. It was also unlikely that sufficient variants could be constructed to satisfy all the requirements, and owing to the specialized nature of the development process it is not possible for users to modify the material themselves (a restriction which also, of course, pertains to the use of a textbook). The designers of the courseware sought to bypass this problem by adopting a strategy of very fine-grained modularity. The material would be divided into very small modules, each corresponding to only a few hours of lectures, which is very much smaller than the one-term granularity of a lecture-based course. At this level of resolution less difference is observed between the teaching at separate institutions. Where a lecturer was satisfied that part of his material was adequately covered by the courseware, he would be able to substitute this for the lectures. Where his own material differed from the courseware, he would continue lecturing as before.

Although our primary objective was to provide an entire suite of computer-based material, we were not unduly concerned that a patchwork of lectures and courseware might develop if that were how the lecturer wished to use it. Indeed, in the event of the courseware being used exclusively we would encourage the introduction of some variation by way of practical work, tutorial contact, written exercises, or occasional lectures on new peripheral topics. The aim was to counteract the inherent dullness of sitting at a computer screen for long periods by frequent changes of activity. We were somewhat anxious about this point because of the minimal supervision of the students, and the high probability that, left completely unsupervised, the less well-motivated may be tempted to hurry through the computer-based material with little attention.

Thus the syllabus was divided into themes, and each theme into a set of modules. The first theme to be addressed covered the area of digital electronics. This was divided into 17 modules, of which the first to be completed comprised a basic introduction to the operation of a CPU. Entitled "Computer Architecture," the module comprised 90 min of material corresponding to 4 h of lectures, and was used as an experiment to evaluate the effectiveness of this type of teaching.

The material we were dealing with was detailed and factual in nature, and not intuitively appealing to many students. A number of techniques which had been successfully employed in the original lecture course were also used in the courseware to maximize the chances that the latter would work successfully with minimal supervision. One important factor was the use of varying styles of reasoning. An inductive approach was commonly used in the earlier stages of an explanation. Here, we proceeded by prompting the student to recall some well-understood fact, then drew an obvious conclusion about the principle we were trying to illustrate. Often we used an explicit question-and-answer format in consecutive frames, for example:

Question:	What does a computer do?
Answer:	It inputs information, stores it internally, processes it, then outputs the results.
Conclusion:	So a computer must have four blocks of electronic circuitry: to input, store, process, and output the information, respectively. They must be connected so as to be able to pass information to each other.



Please click on the control signals which must be activated in order to execute the current instruction in the Instruction Register (IR):

Fig. 1. How instructions are executed.

This leads directly to the basic model of a computer system comprising input, memory, arithmetic unit, and output; a model which is used to inculcate the concept of processing throughout the early part of this module. It is accompanied by animated displays in which the concept of a machine instruction is introduced naturally by showing data moving around the system in response to simple high-level-language commands. The student is then prompted to realize that the machine instructions themselves must have some physical embodiment, and from this is developed the idea of the fetch–execute cycle, control unit, and attendant hardware items such as the instruction register and program counter.

As the complexity increases, a more tutorial style is used to present the less intuitive aspects; for example, the student is directly told how the hardware units are organized internally. In any case, a deliberate attempt is made to juxtapose contrasting styles of presentation, another commonly used technique being an example-driven format.

High degrees of animation and interaction are of crucial importance to the presentation. Animation is a particularly valuable technique when it illustrates a point which cannot effectively be described with a static lecture slide. Interaction is used throughout to maintain the students' interest and to act as a check on too rapid progression through the material without proper understanding. An example is shown in Fig. 1. Here the student is asked to point to the electrical signals which will be activated as a particular instruction is executed. When the correct signals are identified, the student is rewarded with a moving display of the execution of this instruction. A development package, Authorware Professional [9], was specially chosen for its well-developed animation facilities. It offers precise control over visual composition, timing, and trajectories of moving objects, and allows flowchart-based sequencing controlled by accurate positioning of the on-screen cursor via the mouse.

By superimposing the factual material on an undercurrent of continuously changing pace, presentation, and activity, we hoped to maintain the students' interest throughout a computer session as well as we would in a lecture. In the next section we describe how we put our objectives to the test.

III. Experimental Procedure

Broadly speaking, we have set out to answer the following six questions.

1) Do students learn as well from the courseware as from the lectures?
2) If so, then how much time is saved by the lecturer and by the students?
3) Does the courseware have ancillary implementation requirements (supervision, etc.)?
4) Do the students find the work interesting?
5) How much development time is required?

6) How can the course be redesigned around the courseware in order to use the time gained in 2) to the best advantage?

To address the first four issues, we ran a trial of the courseware with a set of specialist Electronic Engineering first-year undergraduates, doing Computer Engineering as a compulsory subject. The course consisted of 13 h of lectures, covering the areas of Number Systems, Computer Architecture, and Assembly Language Programming. The entire class came to the lectures on Number Systems. For Computer Architecture, however, the class was divided into two halves by random allocation. One half spent 90 min (in separate 60- and 30-min sessions) doing the self-learning work, and the other continued to attend the lectures, which for this component amounted to four 1-h sessions. It was emphasized to the students that this part of the course would not be the subject of any official examination, although it did serve as a background to later parts of the course which would be assessed. It was promised that if large discrepancies appeared between the two halves, remedial teaching would be offered to the disadvantaged group.

Afterwards, a short multiple-choice test on the Computer Architecture material was given to the entire group. The test was used to determine whether any significant difference could be observed between the results obtained from the two teaching methods. We also thought it useful to a limited extent to canvass the students' opinions, and while we did not find it acceptable to ask for their judgement on the quality or content of the material, we did seek their views on the preferred style of presentation. From attendance records we also calculated the average time spent in each of the two teaching methods. The latter are simply the times that the students were present either using CAL or attending lectures, and since there were no other materials, for example, notes, available to the students outside that time, it represents a measure of the different delivery times of the teaching.

The two halves of the class were then reunited, and teaching continued in the lecture theater with the next stage of the Computer Engineering course, covering Assembly Language. This was accompanied by intensive practical work in the microprocessor programming laboratory. The students' comprehension of the entire Computer Engineering course was later examined by way of an assignment which involved writing an Assembly Language program to solve a simple problem. The Computer Architecture work served as an essential background to this assignment, while not being directly the subject of it. We have again used these marks to look for a significant difference between the two groups. The analysis in this case also took into account each student's overall end-of-year average, in order to eliminate the effects of any differences in basic ability.

IV. Results

We begin with the results of the multiple-choice test taken immediately after the end of the courseware sessions. Table I shows the results for each teaching method. Column 2 gives the number of students involved; column 3 the mean score obtained out of a total mark of 20; column 4 the standard error of the mean; and column 5 the time spent in hours. Column 6 gives the result of a Student's t-test [10], testing the hypothesis that the results for each teaching method come from the same population, i.e., that there is no difference between the effectiveness of the two methods in producing examination scores. This result is expressed as a "probability value." P values below 0.05 are taken to indicate a reasonable degree of certainty that there is a significant fundamental difference between the two teaching methods. A P value above 0.05 indicates that such difference as was observed could easily have occurred by chance, and that there is therefore no reason to suspect an underlying significant difference. The ratio of times spent by the students, $1.47/3.58$ hours, is $1/2.44 = 0.41$.

TABLE I
1993–1994 Multiple-Choice Test Results

Teaching Method	No of Students	Mean	SEM	Time (h)	P
Lectures	25	12.5	0.66	3.58	
CAL	23	11.4	0.60	1.47	0.21

The end-of-year results are as shown in Table II. One outlier was eliminated from each group. The table shows the final Computer Engineering mark, and the overall end-of-year average excluding Computer Engineering. It is evident from the results that a) the marks for Computer Engineering were higher than the overall averages and b) the overall averages for the students doing CAL were less than of those coming to the lectures. This implies that the CAL group was intrinsically weaker than the lecture group, and that it is therefore not safe to make a direct comparison between the Computer Engineering marks of the two groups. A final set of results was therefore calculated, based, for each teaching method, on the difference between each student's Computer Engineering mark and his or her overall end-of-year average excluding Computer Engineering. It is reasonable to expect that if the CAL and lecturing were equally effective teaching methods, then no significant difference would be observed between these two distributions; this, in effect, correcting the results for differences in basic ability.

In the survey of students' opinions, a majority of 80% expressed a preference for a mixture of lectures and CAL, with 63% preferring a predominance of CAL. 84% would have liked printed backup material, but only 26% wanted people at the courseware sessions to answer questions. Asked whether they found the work boring, 70% said no, 20% slightly, and 10% yes. 45% described the work as easy. When invited to suggest any additional features apart from printed backup, most respondents asked for more interaction and self-check exercises.

It must be mentioned that a previous trial of the courseware was carried out under less well-controlled conditions in which the students were allowed to do the courseware in their own time with no record kept of their attendance. Analysis of the multiple-choice test in this case yielded a P value of 0.0017, indicating a highly significant difference between the performance of the two groups, and led us to impose a certain

TABLE II
1993–1994 END-OF-YEAR EXAMINATION RESULTS

Teaching Method	No of Students	Comp Eng %		Overall %		Comp Eng - Overall %		
		Mean	SEM	Mean	SEM	Mean	SEM	P
Lectures	22	63.0	2.0	60.2	2.3	2.8	2.7	
CAL	24	57.5	1.6	50.6	2.5	6.9	2.3	0.25

amount of external discipline (attendance rota and register check by a lab. assistant) in the trial reported here. We will comment further on this point in Section V.

The time spent developing the courseware (point 5 in Section III) is an important factor. However, the time required to produce this first module, which amounted to several man-years, is by no means representative of the time needed for subsequent modules. A great deal of learning, experimentation, and groundwork had to be done over several participating institutions, and several versions of the material were produced before it was deemed to be satisfactory.

V. Intermediate Discussion and Conclusions

We begin with the results of the multiple-choice test taken immediately after the courseware sessions, as shown in Table I. It is evident from the high P value derived from the t-test that there is no significant difference between the exam scores produced by the two teaching methods, which gives some grounds for optimism that carefully designed courseware can indeed be an effective replacement for lecturing.

Turning to the end-of-year results in Table II, we first observed that the students doing the CAL were weaker overall than those coming to the lectures. We therefore calculated, for each teaching method, a distribution of the difference between each student's Computer Engineering mark and his or her overall average. We then calculated the probability value of a significant difference between these two distributions. It is evident from this P value that once again there is no reason to suppose that the two teaching methods are significantly different. It must, of course, be pointed out that if a similar difference were observed with a larger sample, it would become significant. However, it is at least encouraging that such difference as was observed was in favor of the students doing the CAL. The Computer Engineering marks of the latter were 6.9% higher than their overall marks, whereas those of the students who came to the lectures were only 2.8% higher.

There were two problems associated with this trial. First, the students were aware of its objectives, and of the fact that each group received different teaching. Second, it was not possible to prevent contact between individual groups in order to preserve independence. Both of these factors could have influenced the outcome, but it would have been difficult to devise a trial in which samples from the same population could have been isolated. For example, if we had arranged for two separate institutions to run the trial, each group would still probably have been aware of the objectives, and samples of the performance of each group would have been incomparable because of different admission policies at the two sites. However, it is our view that neither of these problems would have been likely to affect the outcome to any significant extent.

One very striking point is the large reduction in teaching time. The 4 h of lecturer's time were eliminated completely, and that of the students reduced by a factor of about 2.5. It is possibly in this area that CAL will most prove its worth.

Our policy of mixing CAL and lecturing seems to have struck the right note with the students. We appear to have been successful in maintaining their interest, and they seem to have been able to work independently, although a check on attendance was certainly necessary. The students' perception of this computer-based material seems to have been a little more optimistic than has been indicated by some previous studies, for example [6], but the fact that our experiment was not so successful when the students were allowed to work in their own time is a striking analogy with the latter authors' finding that the students need to be compelled to use the system and are unlikely to do so voluntarily.

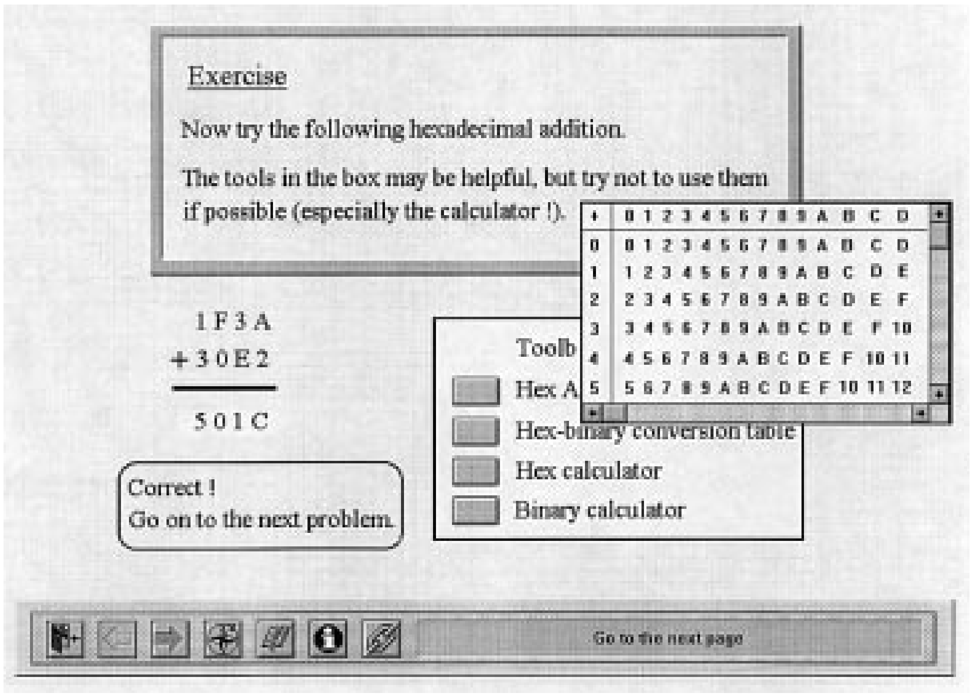
However, the matter of how extensively this type of teaching material can be deployed before the students begin to lose interest is very much open to question. Our survey indicated that they would prefer a predominance of courseware, but there must be some doubt as to whether this preference would be maintained once the "novelty value" had worn off. Related to this is the final issue mentioned in Section III: the question of how best to redesign the course around the new computer-based material. The following section describes a new program to test this point further.

VI. Further Courseware

Given the apparent success of the 1994 trial, our entire Computer Engineering course has been redesigned around three CAL modules. Work started on the two new modules, "Number Systems" and "68000 Assembly Language Programming" in 1996.

It is clear from the trial results that the students appreciate variety in presentational style and activity, and particular emphasis is placed on this aspect in the overall design. More attention has been given, for example, to the inclusion of self-check sequences, an example of which is shown in Fig. 2 from the module on Number Systems.

The students' preference for interaction was addressed in a different way in the Assembly Language module. This is associated with several hours of practical work on M68000 development systems with a debugging monitor. In many Computer Science courses, this work is now done using a simulator such as those described in the references. Consideration was given to interfacing the courseware with one of these simulators, which would have eliminated some hours of


Exercise

Now try the following hexadecimal addition.

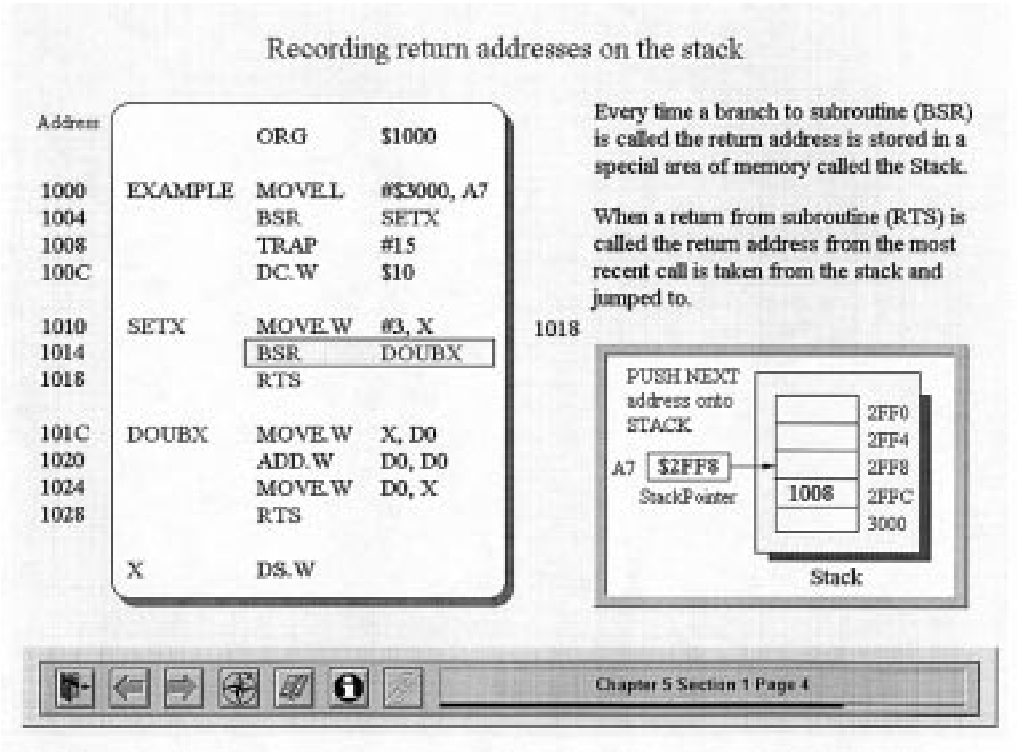
The tools in the box may be helpful, but try not to use them if possible (especially the calculator !).

```
  1F3A
+ 30E2
------
  501C
```

Correct !
Go on to the next problem.

+	0	1	2	3	4	5	6	7	8	9	A	B	C	D
0	0	1	2	3	4	5	6	7	8	9	A	B	C	D
1	1	2	3	4	5	6	7	8	9	A	B	C	D	E
2	2	3	4	5	6	7	8	9	A	B	C	D	E	F
3	3	4	5	6	7	8	9	A	B	C	D	E	F	10
4	4	5	6	7	8	9	A	B	C	D	E	F	10	11
5	5	6	7	8	9	A	B	C	D	E	F	10	11	12

Fig. 2. Hexadecimal arithmetic.


Recording return addresses on the stack

```
Address
                        ORG        $1000

1000    EXAMPLE         MOVE.L     #$3000, A7
1004                    BSR        SETX
1008                    TRAP       #15
100C                    DC.W       $10

1010    SETX            MOVE.W     #3, X
1014                    BSR        DOUBX
1018                    RTS

101C    DOUBX           MOVE.W     X, D0
1020                    ADD.W      D0, D0
1024                    MOVE.W     D0, X
1028                    RTS

        X               DS.W
```

Every time a branch to subroutine (BSR) is called the return address is stored in a special area of memory called the Stack.

When a return from subroutine (RTS) is called the return address from the most recent call is taken from the stack and jumped to.

Fig. 3. The stack.

familiarization time and a great deal of hardware. However, we chose to maintain the use of the real hardware because the courseware is also being used by Electrical Engineering students, who require a practical knowledge of microprocessor development systems in later years. Since this practical work is entirely interactive, a balance was maintained by making the associated courseware more tutorial in nature. This module, like the lecture course from which it was derived, is also dependent on book work. It teaches by example only, and detailed information about the operation of each instruction has to be obtained by the students directly from reference texts during the practical sessions.

An example from this module is given in Fig. 3. Like Fig. 1, this demonstrates the use of an animated sequence to illustrate

TABLE III
END-OF-YEAR EXAMINATION RESULTS IN OTHER YEARS

Year	CAL Modules Used	No of Students	Comp Eng - Overall %	
			Mean	SEM
1992-93	-	40	3.5	2.0
1994-95	CA	51	-12.6	2.2
1995-96	CA, ALP	25	-4.3	3.9
1996-97	NS, CA, ALP	35	-4.6	2.2

TABLE IV
CONTENT OF LECTURE VERSUS CAL-BASED COURSES

Module	Lecture-Based Course (h)		CAL-Based Course (h)		
	Lectures	Practicals	CAL	Lectures	Practicals
Introduction				1	
Num. Systems	4		1.5		
Computer Arch.	4		1.5		
Assembly Lang.	5	15	3		21
Program. Style				1	
Subtotal	13	15	6	2	21
Total	28		29		

a point which is difficult to explain in a lecture, in this case the mechanization of a series of nested subroutine calls and returns. Each involves the saving and restoring of a return address via the stack, with the stack pointer being adjusted at each operation. Fuller, animated examples are available by reference to the FTP site mentioned in the first footnote to this paper.

A course book accompanying each module includes more detailed technical information and further written and practical work. It deliberately does not include more obvious content such as a statement of the objectives or a summary of key points. Despite the students' preference for printed literature, we have some doubts about the value of such repetitious material, as we do not see any difference in principle between asking the students to take notes as they use the courseware and their doing so at a lecture.

The two new courseware modules were written much more quickly than the first one. Much of the underlying framework could be reused, and experience in devising appropriate subject matter proved invaluable in expediting rapid development. The two modules together required about nine man-months of work.

VII. EXPERIENCE OF USE AND DISCUSSION

During the 1994–1995 academic year, the entire Computer Engineering class used the Computer Architecture module; the remaining parts of the course, Number Systems and 68000 Assembly Language, having been taught by lecture. During 1995–1996 the Assembly Language module was used for the first time, and during 1996–1997 the Number Systems module was also added. The end-of-year results for these three years, shown as the difference between each student's Computer Engineering mark and overall average excluding Computer Engineering, are summarized in Table III. Included for comparison is the result for 1992–1993, which was taught entirely by lecture, so that Tables II and III together give a picture of developments over the last five years. One outlier was eliminated from the 1995–1996 group and five (who were the subject of special circumstances) from 1992 to 1993. Any test for significant differences among these results would be of little validity here, as each year was necessarily given different assignment questions, and a major change in examination regulations in 1994–1995 relaxed the requirement to pass this subject at the first attempt. With the exception of this year, it is clear that there is a broad consistency over the five years, with the marks for all years being within a few percent of the average.

At the time of writing (Spring 1998) the three modules are being used together for the second year. Although some of these individual courseware modules are now also entering service as adjuncts to lecture-based courses in many U.K. universities, this is, so far as we know, the first Electrical Engineering degree-level module in which the basic lecturing content has been entirely replaced by CAL activity. The old and new style course contents are summarized in Table IV. It is evident from the table that 13 h of lecturing time have been eliminated, and replaced by 6 h of courseware. We have used some of this saving to introduce one or two subsidiary lecture topics, for example, on programming style, which were not included previously. The aim was to meet the students' preference for at least the occasional traditional lecture throughout the course, and it also has the advantage that the lecturer is allowed to concentrate on more abstract concepts while the students deal with basic matters themselves.

However, the main use to which the extra time has been put is the introduction of many more practical sessions, in which the lecturer can interact at a personal tutorial level with small groups of students. The weaker students gain additional remedial support, and the stronger, well-motivated ones can be introduced to more advanced ideas which would not be appropriate to the general level of the class. The lecturer is thereby relieved of the role of knowledge provider, and in the time saved can fulfil the far more effective role of guide. This, incidentally, seems very reminiscent of the ancient order of university teaching, rarely seen in technical subjects and uncommon elsewhere, but which may now be about to make a welcome return.

Acknowledgment

This work was undertaken as part of the Teaching and Learning Technology Programme, in collaboration with several United Kingdom higher education institutions. We are also indebted to Prof. D. Reynolds of the Department of Education at Newcastle upon Tyne University, for his many helpful suggestions and comments during the progress of this work.

References

[1] *IEEE Trans. Educ.* (Special Issue on Computation and Computers in Electrical Engineering Education), vol. 36, Feb. 1993.
[2] H. B. Diab and I. Demashkieh, "A computer-aided teaching package for microprocessor systems education," *IEEE Trans. Educ.*, vol. 34, pp. 179–183, 1991.
[3] B. L. Barnett, "A visual simulator for a simple machine and assembly language," in *Proc. 26th ACM SIGCSE Tech. Symp. Computer Science Education*, 1995, pp. 233–237.
[4] B. Oakley, "A virtual classroom approach to teaching circuit analysis," *IEEE Trans. Educ.*, vol. 39, pp. 287–296, 1996.
[5] C. T. Sun and C. Chou, "Experiencing CORAL: Design and implementation of distant cooperative learning," *IEEE Trans. Educ.*, vol. 39, pp. 357–366, 1996.
[6] C. A. Cañizares and Z. T. Faur, "Advantages and disadvantages of using various computer tools in electrical engineering courses," *IEEE Trans. Educ.*, vol. 40, pp. 166–171, 1997.
[7] L. P. Huelsman, "Personal computers in electrical and computer engineering: Education survey," *IEEE Trans. Educ.*, vol. 34, pp. 175–178, 1991.
[8] N. Hammond *et al.*, "Blocks to the effective use of information technology in higher education," *Computers and Education*, vol. 18, pp. 155–162, 1992.
[9] "Authorware professional," *Macromedia*, California.
[10] G. M. Clarke and D. Cooke, *A Basic Course in Statistics*. Edward Arnold, 1983.

J. Nicholas Coleman received the B.A. (Hons.) degree in music from the University of York, York, U.K. Some time later he entered Brunel University, Uxbridge, U.K., to work on the Ph.D. degree in the area of CPU architectures for VLSI design acceleration.

In the interim, after working for a short time in commercial applications and systems programming, he joined Plessey Telecommunications Research Ltd., Poole, initially as a Software Engineer but later as a hardware Design Engineer. Here he worked on the specification and design of a special-purpose CPU for real-time production test of the System-X telephone exchange. On graduation from Brunel University, he joined the Department of Electrical and Electronic Engineering at the University of Newcastle upon Tyne. He lectures in computer and digital engineering, and is a member of the university's court and senate. His research interests and publications are in the areas of special-purpose CPU architectures, dataflow computers, computer arithmetic, and computer-aided learning.

David J. Kinniment (M'67) received the M.Sc. and Ph.D. degrees from Manchester University, Manchester, U.K., in 1962 and 1968, respectively.

He currently holds the post of Professor of Electronics in the University of Newcastle upon Tyne, Newcastle upon Tyne, U.K. He has made contributions to several textbooks, and is the author of over 70 papers in the area of electronics design. He has served on panels accrediting undergraduate and postgraduate U.K. electrical engineering education. He has been involved with the organizing committees of a number of conferences and symposia, including the European Design and Test Conference, and the European Workshop on Microelectronics Education.

Francis P. Burns received the B.Eng. (Hons.) and Ph.D. degrees in electronics from the University of Newcastle upon Tyne, Newcastle upon Tyne, U.K.

He worked there for three years in the Department of Electrical and Electronic Engineering as a Research Associate, integrating computer-aided learning material into the undergraduate curriculum in the areas of computer engineering and design tools. He currently holds the post of Research Associate in the Department of Computing Science, working on the TIMBRE (Time-Predictable Hardware Platforms) project. Publications include papers on formal methods, VLSI, high-level synthesis, simulation and CAL. He maintains his interest in, and links with, the CAL project.

Timothy J. Butler received the B.A. (Hons.) degree in mathematics and philosophy from University College Dublin, Dublin, Ireland, in 1988, and the M.Sc. degree in medical statistics from the University of Newcastle upon Tyne, Newcastle upon Tyne, U.K. in 1989.

He is currently a part-time Senior Research Associate (Statistician) in the Faculty of Medicine at the University of Newcastle upon Tyne, and is also registered here as a medical student. He has published papers in the areas of human nutrition, human physiology, medical sciences, genetics, and immunology.

Mr Butler was elected a Fellow of the Royal Statistical Society in 1990, and was awarded Chartered Statistician in 1997.

Albert M. Koelmans graduated from Groningen University, The Netherlands, in 1983, and received the Ph.D. degree from the University of Newcastle upon Tyne, Newcastle upon Tyne, U.K., in 1995.

He joined the University of Newcastle upon Tyne as a Research Associate in the Department of Electrical and Electronic Engineering and has been a Lecturer in the Department of Computing Science since 1985. He has been working in the field of VLSI design, having published papers on hardware description languages, the VISD encryption chip, graphical representation of a CHDL, and transformational reasoning. He is a principal designer of the Newcastle-made high-level VLSI design environment STRICT. His research interests include CAD for VLSI, design of asynchronous systems using Petri nets, and fault-tolerant VLSI systems.