Computer jobs through training—
A preliminary project report

by M. GRANGER MORGAN, MARY R. MIRABITO, and NORMAN J. DOWN

The University of California at San Diego
La Jolla, California

INTRODUCTION

Job training directed toward the disadvantaged population in the United States has been under way for many decades. Traditionally this training prepared people for lower entry level skilled and semi-skilled jobs such as plumbers' aides, welders, clerks, and secretarial help. Only recently, with the expanding awareness of the significant social inequalities which continue to characterize U.S. society, have large numbers of people begun to realize that job training—for just any old job—is not enough. If training is to have any appreciable impact upon the social stratification that characterizes the employment structure, efforts must be made to find high entry level jobs which are suitable for such special training projects.

Of course job training is not the ideal solution to the problem. Something that might honestly be called a "solution" will not come until the children of the disadvantaged receive a quality primary and secondary education and an equal opportunity for college level training. There are plenty of people working on reforming the school system so that such educational equality will one day be a reality. But despite some progress this task is proving exceedingly difficult. In the meantime there is a whole generation of young people who have not enjoyed the advantages of an improved school system and who are without significant job skills. The question is, can we devise job training programs which will train these young people for a career other than in low entry level jobs?

A number of workers have looked to semi-technical and business computer programming as a high entry level job area in which disadvantaged students could perhaps be trained. They have viewed programming as attractive because it does not require many of the social prerequisites, such as the ability to speak dialect free English or a working familiarity with business world interpersonal relations, that are necessary for most high entry level jobs. The only real prerequisites to training someone as a semi-technical or business programmer are an ability to organize ideas in a logical way and some basic math skills.

A number of workers around the country have recently developed programmer training projects designed for the disadvantaged. These workers have come to the problem with different backgrounds and perspectives. The various projects which have evolved display a rich range of ideas, many of which might never have been tested had the central planning and coordination that is widely advocated by educationalists been applied to this development from the outset. What is now needed is a literature which describes these several efforts in detail so that future workers will not have to rediscover what has so far been learned, but can build on the basis of the experience of others. This paper will describe one of these projects, the University of California at San Diego's project, Computer Jobs Through Training.

The basic approach

Work at UCSD on programming instruction for the disadvantaged began in the summer of 1968 when we offered a course in digital logic and basic FORTRAN programming to a group of high school students who were working on the campus in summer jobs made available through the Neighborhood Youth Corps program. This initial course had no long term job training intent. It was offered strictly as enrichment, as something we thought would be a "good thing to do."

We were surprised by how well the course went, and especially by how exciting the students found the subject matter. We began quickly to see that programming and other aspects of computer science were potentially very useful topic areas for reaching and turning on students who previously had not gotten very interested in formal education.
But while it is easy to get these students "hooked" on programming, the standard teaching methods, particularly the formal lecture situation, are totally inappropriate. Programming is best taught to these students the way modern languages are now being taught. Rather than listening to lectures on the grammar of the language students learn the language by using it. Beginning on the first day, the instructor gives a bit of basic introduction and then writes a simple program. He explains it, but doesn't really expect his explanation to be fully understood. The students copy this program, punch it onto cards, which is a painful process since many have never typed, and after some brief instructions run the job themselves on a small computer. Inevitably there are errors but sooner or later the job runs and you see the first glimmerings of understanding and excitement. In the weeks that follow you build on this basic understanding, slowly enlarging on the student’s repertory until he has a command of most of the language.

The physical facilities

This hands-on approach works, but only if there are adequate computing facilities available for all students to use on a continuing basis. Economically the simplest approach is to take the students to a central training facility. In the compact inner cities of our major urban centers this approach also makes good social sense. But in San Diego, while the Black community is somewhat localized, the Chicano or Mexican American community is spread all across the city in a collection of widely spaced communities. In the early portions of the course, motivation is the single most important consideration—and one good way not to motivate people is to make them sit on a bus for an hour or more every day riding to and from a class.

The prospect of establishing a number of training facilities throughout the community was financially unreasonable. In addition, we were reluctant to choose any one portion of the community for our efforts at the expense of others. The solution we chose was a mobile instructional facility housed in a forty foot trailer truck, which through careful scheduling can simultaneously support up to a half dozen courses at different locations all around the city.

A used forty foot trailer was acquired in the spring of 1969 as a gift from Safeway Foodstores, and with support from the Rosenberg Foundation of San Francisco, Montgomery Ward, and the University, the training facility was constructed in this van during the summer and fall of 1969, Figure 1. Our small project staff was considerably aided in this work by a group of Neighborhood Youth Corps students and UCSD undergraduates, largely from the Black and Chicano communities, who contributed many long hard hours of work at low pay along with much enthusiasm and a number of first rate ideas.

The UCSD Computer Center is in the process of installing a large Burroughs B6500 system. Until sometime in 1971 when that system will be supporting a full complement of remote operations, our hardware in the van consists of a small computer with free standing FORTRAN capability and remote batch COBOL ability. When the B6500 system is in full operation this small machine will probably be replaced by a terminal consisting of a small card reader, a line printer and a teletype.

One important hardware requirement is the ability for students to interact with their program during execution. We stress this kind of programming in the early portion of the course because it helps significantly to motivate students and keep interest high. We find too that a drum plotter is a very useful device. Students
work up cartoons and other line drawings with considerable enthusiasm, and the systematic operations involved in pen control make for good practice in step by step logic.

The floor plan in this facility is completely flexible. This results in part from the admissions policy we have adopted. The problem of identifying potentially successful programmers even among college graduates is substantial. Making this identification for disadvantaged young adults is an almost impossible task. It is widely recognized that aptitude tests display a cultural bias. More importantly, since many of the students we hope to reach “turn off” in a testing situation, we feel that massive pre-testing, which has been the approach of some experimental programs, is not the answer.

Obviously we must require basic math and logic skills, and a level of intellectual development on the part of our students roughly equivalent to that of a high school graduate. We do not specifically require a high school degree, though most of our students have one or are in the process of getting one.

To check for math and logic skills we administer a short entrance test, which like all of our introductory material is bilingual, with English on one side of each page and Spanish on the other. But our basic approach to entrance has been—anyone who seriously claims he wants to be a programmer may enter the course. If he has not done well on the entrance exam we warn him that he will have trouble. But no one who is really serious in claiming that he wants to take the course has been excluded. Actual performance during the first weeks of the course is the real entrance test.

All this is fine, but one must be realistic. Many students will quickly discover that programming is just not “their bag” and will drop out of the course in the early weeks, others will stay with the course for some while, but despite good motivation will just not be able to do the work. These latter students we are trying to direct towards alternative more appropriate forms of training, such as the San Diego Urban League’s key punch school, so that their CJTT experience will not represent a failure, but rather a first step toward something else.

Given the diminishing class size which results from this approach to admissions we have designed the van so that we can start out accommodating relatively large
numbers of students and then eventually switch over to a more spacious floor plan once the class size has fallen off. In addition, arrangements have been made to allow the van to be subdivided into smaller areas for group work with teachers aides. A laboratory set-up with normal work benches is also possible. Figure 2 shows two interior views of the van.

Curriculum considerations have dictated a number of other aspects of the physical facilities. In our work with Neighborhood Youth Corps students during the summer of 1968 we looked at a large number of the 16mm films on various aspects of computer science which are available from industry. Almost none of these films are suitable for use with our students. They are either much too technical, or much too simple minded. As a consequence a large set of 35mm slides has been developed for use with the course. These slides come in three types: course slides, which directly support the curriculum with flow charts, diagrams, drill exercises and similar materials; computer science orientation slides, which provide students with an introduction to the physical components of computer science, explain how they work, and introduce the student to a large number of typical system applications such as airline reservation systems, medical diagnostic systems, scientific systems, production control systems, and so on; and social orientation slides which consider things like how to act and what to expect in a job interview. So that these slides can be used as an integral and natural part of the course, without disrupting the flow of thought when slides are introduced, the van has been equipped with a remotely controlled projection system and variable intensity lighting. The instructor is able to use slides easily and at his convenience.

Finally, to support the hardware portion of the course which is described below, the necessary DC logic voltages, signal lines, and 110 volt lines are distributed to convenient plug panels located for student use throughout the van from power supplies and signal generators located in a small shop area in the forward portion of the van.

The instructional program

Before describing the details of the curriculum which has been developed for the CJTT course it is important to explain what kind of person, with what kind of programming skills, we are training in this project and what he will most likely do when he finishes the course. Clearly we will not be producing systems programmers. What we will produce are competent coders and programmer/trainees, who unlike the graduates of many private data processing schools will have a solid foundation in the logical aspects of programming. Our graduates will be able to take a well stated word problem, work up the necessary logic, develop the flowchart, produce the necessary code, debug the program, and make it run.

But despite our earlier observation that the only real prerequisite to success as a programmer is an ability to think logically and a command of fundamental mathematics, it is nevertheless important to realize that while many of today's successful programmers worked their way up with only a high school degree, this is becoming increasingly difficult to do. More and more a two year AA in data processing or, better still, a four year BA is becoming prerequisite to substantial progress up the data processing ladder from the lower coder and programmer positions.

Recognizing this, and understanding that we can reasonably expect to train people who at the outset are employable only in the lowest positions, we have attempted to design both our own project, and the kind of job placements we have arranged, in such a way as to maximize the possibilities of further education for our students. Hence in our course we emphasize a strong foundation in the basic logical techniques of programming rather than the sort of cook-book introduction to existing operating systems that is characteristic of many of the private data processing schools. We treat specific programming languages as secondary in importance to the fundamental ideas of program organization. But, in our choice of languages (FORTRAN and COBOL) we have been careful to select those languages which we think have the best potential for immediate employment, consistent with our long term objective of further education.

We begin with FORTRAN. Most of the semi-technical programming jobs in the San Diego area require FORTRAN, as do almost all of the major employers with good programs for continuing employee training and education. FORTRAN has two other important advantages. Unlike COBOL, which requires a substantial knowledge of syntax before even simple programs can be written, it is possible for students to run and understand simple FORTRAN programs on the first day of class. A second advantage is the easy use of subroutines, an aspect which we consider essential in teaching basic programming concepts.

The basic curriculum for the CJTT project was evolved during the summer of 1969 with the support of the Rosenberg Foundation and received preliminary field testing on a second group of 15 Neighborhood Youth Corps students in an 8 week summer course at UCSD, Figure 3. The organization of the course is shown in Figure 4.

The formally developed curriculum material con-
Figure 3—Neighborhood Youth Corps students from the eight week pilot course run during the summer of 1969

consists of a carefully graduated sequence of problems designed to be as familiar and interesting to the student as possible. The order of presentation of basic FORTRAN instruction in the first third of the course is:

I. Simple Math
II. Loops and Sorting Using the Computed GO TO
III. IF Statements
IV. Data Arrays and Alphameric Formats
V. Subroutines and Special Math Methods
VI. DO Loops

Because of the great significance that we attach to getting our students fully versed in all aspects of program logic, DO loops are purposely not introduced until the very end of the introductory section so that students are forced to manufacture all the looping structures they require. With a very few exceptions, all our problems are presented as word problems, both to force the development of reading skill and also to get students in the practice of working from written instructions, some-

thing most of them have done only rarely in their previous activities.

Along with this graduated difficulty in program logic goes a review of basic math concepts. This review takes place as part of the programming rather than as a separate topic since most of our students have been "turned off" by math in school, largely because they never could see any need or use for math. Having gotten the student hooked on programming, it is then possible to undertake a math review in the context of programming which students would never tolerate as a simple abstract review.

All of the early material in the course is available in English on one side of the page, and Spanish on the other. Clearly no programmer can be placed in a job in this country if he is not fluent in English. Fluency in English is a prerequisite for entrance to the course. But being fluent in English and being comfortable in English are two different things. During the 1969 Youth Corps course we found that several of our Chicano students became much more interested and did much better work when problems were available bi-lingually and when instructors showed a willingness to use Spanish. It is clear that even students who when given a choice frequently use the English version of problems nevertheless appreciate having the Spanish version

Figure 4—Diagram of course curriculum. Time runs vertically
You have saved up enough money to treat yourself to dinner out. If you have $3.00 or more you can go to Sister Paige's; if you have $2.00 or less, you can go to Sullivan's Barbecue for chicken's and peas; if you have $1.00 or less, you can go to the Happy Hut. If you want to eat at McDonald's for a hamburger; if you have less than a dollar, you have to eat at home. Fill in the flowchart below and write a program to do the above.

![Flowchart](image-url)

**Problem 1**

You have been hired to help automate the sales records in a discount store. The sales manager does not know much about computers and programming, so as a sample, you are going to write him a program that will make the computer do what he has a very, very, very difficult time doing.

You want to be able to get the price of a number of items, compute the tax on the taxable items, and then all things. All you need to figure out is how much the customer owes you. You also want to figure out how many green stamps to give him. You want to do this over and over again for each customer.

In addition, you want to keep track of the total tax collected during the day, and the total cash sales in each of the 5 categories.

<table>
<thead>
<tr>
<th>Code</th>
<th>Item</th>
<th>Tax</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>books</td>
<td>none</td>
</tr>
<tr>
<td>2</td>
<td>liquor</td>
<td>15%</td>
</tr>
<tr>
<td>3</td>
<td>stationery</td>
<td>5%</td>
</tr>
<tr>
<td>4</td>
<td>hardware</td>
<td>5%</td>
</tr>
<tr>
<td>5</td>
<td>clothing</td>
<td>5%</td>
</tr>
</tbody>
</table>

Here is a step by step description of how your program should work:

1. **Type in the code and price of each item.**
2. **When you have entered all the items, the program will print out the total purchase price, the tax, and the number of green stamps.**
3. **The program will compute the total tax for the day.**
4. **At the end of the day you will have one stamp for each dollar spent in this store.**

```
IF (NUZ) Z00
   FORMAT ('NUZ IS NEGATIVE')
   GO TO 38
20    WRITE (1, 21)
   FORMAT ('$', F10.2)
   GO TO 10
24    WRITE (1, 25)
   FORMAT ('$', F10.2)
   GO TO 38
25    FORMAT ('NUZ IS ZERO')
30    CONTINUE

IF (NUZ) Z00
   PRINT Z1, Z2, Z3, Z4, Z5
   GO TO 30
```

**Problem 2**

The IF statement is a FORTRAN instruction to alter the operation of the program depending upon the conditions which exist. Here is a simple example using the IF statement: A program computes the value of NOZ. The programmer wants to know the value of NOZ to be negative, zero, or positive. He writes:

```
   IF (NUZ) 20, 24, 28
   WRITE (1, 21)
   FORMAT ('$', F10.2)
   GO TO 10
```

In this case if NOZ is negative the program jumps to 24; if it is zero it jumps to 28; if it is positive it jumps to 28. In place of NOZ you can put some expression like:

```
   IF (NUZ = 0) 21, 22, 24
```

Some friends of yours who are publishing a newspaper for the Black and Chicano community have learned that you are a programmer and have asked you to help them automate the billing and records for their advertisements.

1. Advertisements are sold by the column inch. The rate is $2.50 per column inch for the first 5 inches, $2.00 per column inch for all space beyond 5 inches. Flowchart and write a program which will read in the rate of an advertisement in column inches from the keyboard and type out the price. Write the program so that it will continue to keep back and read in new numbers as long as you wish.

```
   FORMAT ('$', F10.2)
   GO TO 10
```

**Figure 5—Typical problems from the early stages of the course. All early problems are available in English on one side of the page and Spanish on the other.**
available and feel more relaxed in the course as a consequence.

A few examples of some of the early problems for the CITT curriculum are given in Figure 5. Along with these problems, brief non-credit drill sheets are used massively in the early portions of the course. Unfortunately it is not possible in the written version of this paper to give a proper impression of the 35mm slides which have been developed to accompany the course. This will be attempted in the oral version of the paper.

A knowledge of digital logic is hardly prerequisite to most programming jobs, though it does make some of the fine points of programming more intelligible. But hardware can serve as an excellent motivational tool. We explain to our students that computers are complicated in just the way that a house is complicated. The bricks, nails, and boards which make up a house are conceptually simple. It is only when thousands of them are combined in a building that you end up with something that is complicated. In much the same way, AND-gates, OR-gates and flip-flops are logically simple devices. It is only when thousands are wired together to make a computer that you end up with a complicated system.

We introduce hardware with as little talk about actual circuit elements and abstract symbolism as possible. The AND gate is introduced with a demonstration which solves the problem “if it’s sunny AND your friend can come, you can go to the beach.” The truth table is worked out in terms of English:

<table>
<thead>
<tr>
<th>is sun out?</th>
<th>can friend come?</th>
<th>do you go?</th>
</tr>
</thead>
<tbody>
<tr>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>no</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

Once such a truth table is introduced for several simple problems, the jump to more abstract levels of binary notation is not too difficult. Likewise, more complex circuit configurations such as two AND gates with their outputs ORed together are used to solve day to day problems. For example the circuit:

```
+-------+-------+-------+-------+
| sun/no sun — friend can/can’t come — weekend/not weekend — action/no action at home |
|       |                     |                     |                   |
|       +-----------------+-----------------+-----------------+
| go/don’t go to beach |
```

solves the problem, “it’s sunny AND your friend can come OR if it’s the weekend AND there’s no action at home, you go to the beach”.

In keeping with the hands-on philosophy of the project, individual logic plug boards which will allow each student to work up his own circuits have been developed as shown in Figure 6. The development of this equipment has been made possible with a grant of T-series logic from Xerox Data Systems.

To conclude the hardware unit there is a final class project. In an introductory class for engineers, the class might build tape controllers, parallel to serial converters, or similar devices. Many of our students find such examples rather unexciting. Instead, we have chosen a problem which is technically just as demanding, but which is substantially less abstract. A model N-guage rapid transit system for the city of San Diego has been built and outfitted with appropriate micro-switches which provide information on the trains at all times. The set up is shown in Figure 7. The class is asked as a group to develop the control logic necessary to automate the system.

With this hardware background, an introduction to machine organization and the fundamental ideas of as-
The tab equipment unit is designed to give students a very brief introduction—it is not designed to train experts. We quickly outline the use of the program drum on the keypunch and then briefly introduce and use the sorter and accounting machine. The project has acquired 62 surplus type 910 control boards so that each student is able to wire one simple listing problem.

As indicated in Figure 4 the middle third of the formal portion of the course is devoted to more advanced programming concepts such as file organization and maintenance, extensive use of subroutines, and similar techniques which are required for simple system work. The final third of the formal curriculum introduces COBOL. This portion of the course involves few new logical ideas. Indeed, students do many of the same problems that they have already worked in FORTRAN, so that they can develop a feel for the comparative strengths of the two languages.

Well before the end of the course it is clear which students are doing well enough for job placement. When these students complete the course they enter a Terminal Workshop of intensive full-time training for a period of about seven weeks which prepares them for their job. If the potential employer has indicated that he requires specific skills, such as COBOL proficiency or extensive magnetic tape experience, these are emphasized. The Terminal Workshop is run at the UCSD Computer Center and at nearby Gulf General Atomic. Though we have serious financial difficulties, it is our intention that all students who require financial assistance during this full time terminal workshop period will receive a stipend.

Job placement

It is not sufficient in a project such as this one to count simple placement in a programming job as success. What counts of course is the number of people who continue to work in the field long after initial placement. In order to minimize difficulties in the early months after placement our students will be followed carefully once they are out on the job by a Placement Aide who will try to detect difficulties, either of a social or technical nature, well before they become serious, and take the necessary corrective action.

Our first course for young adults got under way in February 1970 on a nighttime basis. We made arrangements with local employers for placement of the modest number of graduates that we expected from this first course before the course began. However it is not reasonable to expect to get massive commitments for placement for many students before a project such as ours has produced its first graduates. Programmers are not plumbers’ aides. While industry will commit itself to hire large numbers of low entry level people from training programs, any reasonable employer will insist on seeing the quality of the graduate before he commits himself to hiring someone like a programmer.

To sell the project to employers we are using the mobile aspect of our facility—setting up in a potential employer’s parking lot with the van and a number of our students and asking the employer to come have a look at what we are doing. We are also seeking employers’ commitments to hire our graduates through local organizations such as the Urban Coalition. At the time of this writing the first full adult class has not yet been graduated. Further details on the placement aspects of the project will be provided in the oral version of the paper.
Other aspects of the CJTT program

While job training is the principal objective of the CJTT project, the two Youth Corps classes that we ran during the summers of 1968 and 1969 were valuable in their own right as courses which stimulated and motivated disadvantaged high school students toward further education and careers in computer science or in other technical fields. We have been operating with a small staff and a very small budget, and this together with the fact that our primary objective has been organizing the job training project has prevented a careful systematic follow-up on all of the NYC students. However, judging from those students with whom we have maintained contact, the courses have had a significant impact.

During the summer of 1970 we have made more formal arrangements to continue this motivational type of instruction for high school students. With financial support from the San Diego Unified Schools we are running three special motivational classes for credit as part of the city school’s summer session. These classes, two for high school students, one for entering seventh graders, are being conducted with very high enrollments of disadvantaged youths. The instruction is being done by four UCSD Black and Chicano undergraduates who are majoring in computer science. Progress in these classes has been excellent and we expect to expand our in-school activities.

CONCLUSION

We have restricted ourselves in this paper to a straightforward description of the CJTT project but it would not be proper to conclude without giving some indication of the very serious financial difficulties that we have encountered. On the local UCSD campus we have received strong moral and financial support from Professor Kenneth L. Bowles who directs the Computer Center and Professor Henry G. Booker, Chairman of the Department of Applied Physics and Information Science. Outside of this support, which had totaled just under $40,000 by July of 1970, we have pieced together an additional $40,000 from dozens of separate sources, most of which are listed in the acknowledgments.

Much of this $80,000 of support which had been organized as of July 1970 was in-kind assistance. With this support the CJTT project has accomplished what by normal University operating methods would have cost slightly more than $150,000. This savings did not come easily. It results from substituting labor for capital. It was accomplished at times by turning highly qualified programmers into painters and carpenters; by scrounging used electrical conduit from old buildings about to be razed; scrounging lumber from construction site foremen; putting Ph.D.s to work doing carpentry and digging telephone pole holes.

Despite more than a year of strenuous salesmanship and much proposal writing, the project had still not received any State or Federal anti-poverty monies as of July 1970. Our evaluation of the funding situation is that this experience is not unique, that others planning similar programs can anticipate similar very serious funding difficulties unless they can find strong sources of local or private financial support.

Our estimated costs are roughly $150,000 per year or just over $3,000 per student placed. Something like 80 percent of this cost is for salaries. No student stipend costs are included in these figures. While we began this work with great optimism, our experience to date has led us to seriously question whether long term funding of this magnitude can be organized. There is much talk in the country about how important it is to do this sort of thing but not very much money to do it. Future workers would do well to explore the financial climate with great care before launching new projects.

REFERENCES

1 While a number of workers have undertaken projects in this field, the literature is still very spotty. A review of some of these projects is available in:

D R MAYER
The involved generation—Computing people and the disadvantaged
AAPS Proceedings of the Fall Joint Computer Conference 1969 p 679

In addition to the work reviewed in Mayer’s paper we are aware of work undertaken by:

R BELLMAN J BLOOD C FORD-LIVENE
Project Soul: Computer training for high school students from disadvantaged areas
University of Southern California Technical Report UCSEE-375 November 1969

T I BARTHA
Computer programmer training program
Report of the Computer Education and Research Center Pratt Institute Brooklyn New York 11205

L H HARRIS
of Shell Development Corporation (P. O. Box 481) in Houston, Texas, has run a training program for a number of years.

ACKNOWLEDGMENTS

We wish to acknowledge the invaluable assistance of Kenneth Bowles, Henry Booker, Jack Douglass, Frank
Saiz, Roy Cazares, Ken Hicks, Oneeta Alexander and Bob Sadler of the UCSD staff, along with the assistance of many UCSD undergraduates and Neighborhood Youth Corps students of whom Belton Flournoy, Susan Halfon, Calvin Manson, Beverly Andrews, and Boyd Pearson deserve special mention.

In addition, we gratefully acknowledge the financial assistance of the University of California, The Rosenberg Foundation, Gulf General Atomic, Safeway Stores, Xerox Data Systems, Montgomery Ward, Bekins Van Lines, Pacific Gas and Electric, Pacific Telephone, and other local supporters.