1620 Users Group
Joint Canadian/Midwestern Meeting
Pick-Congress Hotel, Chicago, Illinois
March 3, 4, & 5, 1965

PROGRAM

Wednesday, March 3

9:30 AM Opening Remarks

Introductions

"A Survey of Recent Books on Numerical Analysis and Programming."
Charles Davidson, University of Wisconsin.

10:15 AM Coffee

10:45 AM IBM Announcements - New Equipment and Systems.

12:15 PM Lunch

1:30 PM Programming Systems

General American SPS - A high speed assembler for paper tape systems. Mention will also be made of two associated utility routines, a Label Reference Indexer (for tape or cards) and an Editing and Tape Titling System.
Peter Boekhoff, General American Transportation Corp.

CHITRAN - A PDQ based FORTRAN with shorter arithmetic routines, several extensions of the language, and comparable timing.
Peter Boekhoff and John Trantum,
General American Transportation Corp.

1:30 PM IBM Seminar-System/360

A more detailed and technical treatment of the System/360 than was made during the "announcements" session, with opportunity for questions and discussion.

1:30 PM Monitor Workshop-Elementary/Intermediate

A session for the beginner or user who is interested in using the Monitor systems as IBM wrote them.

2:45 PM Coffee

3:15 PM Monitor Workshop-Elementary/Intermediate (continued)

Wednesday, March 3(cont.)

3:15 PM Engineering

A FORTRAN procedure for wind analysis of a building of unlimited size and variable moment of inertia.
James W. Madden, Electronic Data Processing, Inc.

A numerical control program language for the 1620.
Anthony Amort, Beloit Corporation.

3:15 PM Education

The use of mark sensing equipment in the preparation, processing, and execution of student written programs.
H. B. Kerr, Tennessee Tech.

Grade Normalizing and Plotting.
Joyce Foder, University of Wisconsin.

7:00 PM New Users Meeting

An introduction to the 1620 Users Group for those attending their first meeting.

7:30 PM SOUND-OFF

Our chance to speak directly and publicly to IBM concerning their hardware, their software, their policies, and their treatment of their best customers - us. We have been promised an answer to all questions asked, if not at this session then at the answer session on Friday morning.

Thursday, March 4

9:00 AM Monitor Workshop-Advanced

A session for those interested in the internal construction of Monitor, with an eye toward changing it to make it work better.

9:00 AM Demonstration Programs

Demonstration Programs Enjoying Great Popularity.
Kurt Eisemann, Catholic University of America

HOOTIE I & II, a music program for the IBM 1620.
Peter Boekhoff, General American Transportation Corp.

Thursday, March 4(cont.)

9:00 AM Programming Systems

TABTRAN - A language to facilitate, cross-tabulation, and analysis of multiple data cards per entity.
William R. Best, M.D., Biostatistic Research Support Center, Hines V.A. Hospital.

10:15 AM Coffee

10:45 AM Monitor Workshop-Advanced (continued)

10:45 AM IBM Seminar-1620 Drafting System

A more detailed and technical treatment of the 1620/1627 drafting system than was made during the "announcements" session, with opportunity for questions and discussion.

10:45 AM Programming Techniques

Machine Language programming techniques. A tutorial session that assumes familiarity with SPS and machine language at the elementary/intermediate level.
Richard L. Pratt, Data Corporation.

12:15 PM Lunch

1:30 PM "Programming Languages and Where They are Going."
D. D. McCracken

This must be regarded as a "Feature Presentation". Dr. McCracken is as well known for his knowledge of programming language theory as he is for his ability to communicate his ideas.

3:00 PM Coffee

3:30 PM Programming Systems

The 1710 FORTRAN Executive Programming System.
James C. Deck, Inland Steel Company &
Gordon Kaufmann, IBM

The University of Toronto Operating System. An operating system for 1710 users.
E. S. Lee, J. A. A. Field, P. I. P. Boulton
University of Toronto

3:30 PM IBM Seminar-IBM 1130 (D. Gibson)

A more detailed and technical treatment of the 1130 than was made during the "announcements" session, with opportunity for questions and discussion.

Thursday, March 4(cont.)

3:30 PM Applications

A General Traffic Handling Systems Simulator.
Donald L. Dietmeyer, Univeristy of Wisconsin

Linear Multiple Regression with Prescribed Terms.
Richard W. Nelson, Institute of Paper Chemistry

Determination of the Normalized Autocorrelation Function.
James S. Taylor, Data Corporation

Friday, March 5

9:00 AM Programming Systems

PDQ-P FORTRAN. An improved version of PDQ that includes provision for use of the 1443 Printer.
James S. Taylor, Data Corporation

The development of a Syntax Directed Translator, a new approach to a more generalized language for computer program generation.
Robert A. Freiburghouse, University of North Dakota

Modifications to Monitor I.
Fred A. Hatfield, Line Material Industries.

9:00 AM Education

Registration Statistics of the 1620 for the small college.
Arthur F. Jackson, The Agricultural and Technical College, Greensboro, North Carolina.

Speciman Machine Interpreter. A teaching device to allow students to learn machine language programming on a simple hypothetical computer.
T. R. Hoffman, Union College

The 141 Data Processing Machine. An educational device for teaching elements of computer programming.

10:15 AM Coffee

10:45 AM The Answer Man. IBM's answers to the Sound-Off Session and to other questions.

12:15 PM Lunch and formal closing of meeting.

1:15 PM Meeting rooms will be available for special interest groups and/or discussions.

CHITRAN

Peter G. Boekhoff,
John W. Trantum

General American Transportation Corporation

USER #3193

March 3, 1965

INTRODUCTION

CHITRAN is a system based, for reasons which by now should be familiar to most USERS, upon PDQ FORTRAN. The objective of CHITRAN is to retain all the features (including running speed) of PDQ while providing a more powerful source language and a shorter, more efficient object program. In addition, CHITRAN provides several system options (in particular, load-and-go in a 40K or larger machine) not available in PDQ, UTO, or Cro-Magnon FORTRAN, and improved diagnostics and a revised error routine.

CHITRAN does possess one disadvantage as compared to PDQ. It was necessary to come up with a shorter Class A arithmetic package, and there is no tightening PDQ arithmetic; Mr. Holmes was not kind enough to include any slop in his routines. So a new set of routines was written; while considerably shorter, CHITRAN arithmetic is a bit slower. No exact times are available at this writing, but the difference is on the order of two to five percent. The function routines are almost identical to those of PDQ, with the addition of mode checking and improved diagnostics; they are therefore longer and slightly (not perceptibly) slower.

SOURCE LANGUAGE

The CHITRAN language includes PDQ (with REREAD) as a subset (except for DRH), which defines most of the CHITRAN language. This section describes the additional language specifications.

Function subroutines have been expanded. In addition to the SIN, COS, EXP, LOG, SQRT, ATAN, ABS, and DRH (retitled INT) functions, CHITRAN includes SIGN, ONE, FIX, and FLT. SIGN is the signum function also present in UTO; its value is -1, 0, or +1 respectively as its argument is negative, zero, or positive. ONE is the Heaviside unit step function, defined to be 0, .5, or 1 respectively for negative, zero, or positive argument. FIX fixes a floating-point argument; FLT floats a fixed-point argument.

Each function will operate on an argument in either mode, and with the exception of FIX and FLT, each returns a result in the same mode as its argument. This produces some intriguing results; for example

$$\text{COS}\ (I) = \begin{cases} 1, I=0 \\ 0, I\neq 0 \end{cases}$$

$$\text{ATAN}(I) = \begin{cases} -1, I<-1 \\ 0, -1\leq I\leq 1 \\ +1, I>1 \end{cases}$$

FIX and FLT are in fact identical. Each simply reverses the mode of its argument. Thus the FIX-FLT routine is involutory:

```
FIX (3.0) = 3
FLT (3.0) = 3
```

FLT(FLT(3.0)) = FIX(FIX(3.0)) = 3.0.

The ASSIGN statement is perhaps the most powerful (and dangerous) in the language. Its "normal" use is in the sequence

```
ASSIGN 3 TO J
      .
      .
      .
GO TO J
```

which will cause a branch to statement number 3. The ASSIGN statement may, however, literally be used to assign anything to anything. For example,

```
ASSIGN 2.718 TO ARRAY(2,3)
ASSIGN SQRT,TO 3
ASSIGN 3 TO 4
ASSIGN 3, TO 4
ASSIGN A, TO I
```

are all valid. Note that if the first argument is a function name or a non-subscripted variable, it must be followed by a comma. The comma after a statement number, floating constant, or subscripted variable is optional.

ASSIGN simply generates a transmit record instruction, and may therefore be used (intentionally or otherwise) to redefine any part of the symbol table, or all of core. A small amount of thought on the part of the programmer is recommended. Record marks will appear in the symbol table at the end of the branch instruction corresponding to a statement number, at the end of the 20-digit entry corresponding to each Procedure, at the end of the branch for each function subroutine which is used in the program, and in each undefined variable.

A statement-number list in an assigned GO TO will be interpreted by the processor as a subscript list; thus

```
GO TO N(3,4,5,6)
```

will generate a branch to N(3,4) and will result in a compilation error if N has not been dimensioned.

The statement COMPARE A,B will compile the single instruction

```
C  A,B
```

Sense switches 11, 12, or 13 may then be tested. Since excess-fifty notation permits a straight compare of two floating-point numbers, the load-and-subtract of an IF(A-B) statement may be avoided. Since 87.3% of all IF statements are of this form, the programmer who knows how to use the H/P and E/Z indicators can economize considerably. A, B, or both may be subscripted. The only restriction is that A must not be a fixed-point constant (B may be).

The frequent appearance in FORTRAN programs of such statements as PI=3.1415927 and ZERO=0.0 has led to the inclusion of the DATA statement in CHITRAN. Whereas the above two statements will generate 4 symbols and 2 instructions, the statement

```
DATA(PI=3.1415927),(ZERO=0.0)
```

accomplishes the same purpose with only two symbols and no object instructions, by loading the constants into PI and ZERO at load time. With certain restrictions, alphameric data may also be defined this way. The first character must be non-numeric, no more than five (nonblank) characters are permitted, and the data will be assembled left-justified with blanks cleared. Thus the statement

```
DATA ( TITLE = N A M E )
```

will cause $\bar{5}541544500$ to be loaded into location TITLE when the object program is loaded.

The DUMMY statement reserves space in the object program for subsequent patching. We define it here by example. Space for five instructions may be reserved by the statement

```
DUMMY 60
```

which will generate the 7-digit instruction

```
B7 *+60
```

followed by 53 digits of garbage. The branch instruction prevents disaster if the programmer decides (none of us ever forgets) not to include the patch. The number of digits must be an unsigned integer less than 10000. The processor will merrily accept a DUMMY statement with an odd number. Caveat.

As implied in the description of the COMPARE statement, two-digit SENSE SWITCH numbers may be referenced in an IF statement. The IF statement includes a continuation feature: any (or all) of the statement numbers following the right parenthesis may be replaced by *, which refers to the following statement. That statement need not be numbered unless it is referenced elsewhere. For example, the statements

```
      IF ( J-3 ) 1, 2, 1
1     I = I + 1
2      ...
```

may be written

```
      IF ( J - 3 ) *, 2, *
      I = I + 1
2     . . .
```

thus saving two branches and the symbol-table entry for the statement number 1. In this particular example, of course, it would be more efficient to write

```
      COMPARE J, 3
      IF (SENSE SWITCH 12) 2, *
      I = I + 1
    2 . . .
```

Two FORMAT specifications have been added. These are Sw.d and Kw. On input, these are treated respectively as Fw.d and Iw. On output, they are also Fw.d and Iw unless the numbers being output are zero, in which case Sw.d and Kw are both converted to wX. The function of these specifications is thus simply zero suppression.

The MOVE statement appears in the language at the suggestion of the ubiquitous J. W. Holmes. Its basic form is

```
MOVE n,A,B
```

which will transmit n consecutive fields beginning at A to n consecutive locations beginning at B. For example,

```
DIMENSION A(5,3),B(5,3)
. . .
MOVE 5,A(1,1),B(1,2)
```

will move the first column of matrix A to the second column of matrix B, and is equivalent to

```
B(1,2) = A(1,1)
B(2,2) = A(2,1)
B(3,2) = A(3,1)
B(4,2) = A(4,1)
B(5,2) = A(5,1)
```

The MOVE statement may also be written as

```
MOVE n,A,B,i,j
```

This will cause every ith field beginning at A to be moved to every jth location beginning at B. Thus

```
DIMENSION A(5,3),B(5,3)
. . .
MOVE 3,A(1,1),B(2,1),5,5
```

will move the first row of A to the second row of B, and is equivalent to

```
B(2,1) = A(1,1)
B(2,2) = A(1,2)
B(2,3) = A(1,3)
```

Either i or j may be zero; thus

```
DIMENSION A(5,3)
    . . .
MOVE 15,ZERO,A(1,1),0
```

will move ZERO to each element of A (the omitted argument will be assumed to be 1).

Another of the many uses for this statement is the replacement of the sequence

```
  DIMENSION K(6), A(6), B(6), C(6), X(6)
          . . .
  DO 1 I = 1,6
1 READ 100, K(I), A(I), B(I), C(I), X(I)
```

by the sequence

```
  COMMON K(6), A(6), B(6), C(6), X(6)
          . . .
  DO 1 I = 1,6
  READ 100, K(6), A(6), B(6), C(6), X(6)
1 MOVE 5, K(6), K(I), 6,6
```

which will compile five fewer object instructions (saving 60 locations).

The increments i and j may also be negative; thus the first column of A may be reversed by

```
DIMENSION A(5,3), TEMP(5)
    . . .
MOVE 5,A(1,1),TEMP(1)
MOVE 5,TEMP(1),A(5,1),,-1
```

The MOVE statement generates 3 instructions in the object program. The restrictions are that only one of the arguments A and B may contain variable subscripts (this restriction will be removed if compiler space permits), and that n, i, and j must be fixed-point constants, signed or unsigned, between ±999. If $n \leq 0$, no data will be moved.

If any operand is omitted, its omission must be indicated by a comma. The statement

```
MOVE, P, Q,,3
```

is a valid way to write

```
MOVE 0,P,Q,1,3
```

since n is assumed to be zero if unspecified; i and j, to be one.

DIAGNOSTICS AND DEBUGGING

Compilation diagnostics are for the most part the same as those of PDQ, with additions related to the new statements. Two further diagnostics have been added. An expression of the form I**J, which used to be accepted and to compile garbage, will result in CHITRAN in an error #5 (same as mixed mode). An error will be generated by the END statement if there are any DO statements whose ranges have not been defined.

The error routine has been rewritten. Object-program punching is no longer terminated when an error is found. Instead, the appropriate error message is typed, the computer halts, and one of three actions may be taken:

1) Get off the machine.

2) If switch 2 is on when START is pressed, a corrected statement may be entered at the typewriter.

3) If switch 2 is off, the erroneous statement is ignored and processing continues.

Execution diagnostic procedures have been modified. The major modifications are in SIN-COS and in LOG. If the argument in SIN or COS is too large, an error message is typed and the result set to zero, which seems a bit more productive than an arbitrary dead-end halt. If the argument in LOG is zero, the result is set to $\bar{9}999999999$ (minus infinity), instead of the nonsensical $\bar{9}999999999$ (plus infinity). Arguments out of range in EXP will yield the messages EXOFLO or EXUFLO, according to the sign of the argument.

To facilitate debugging, $\bar{6}9\bar{0}000\bar{\neq}000$ will be placed at load time in each symbol-table location whose contents have not been defined. Any attempt to use an undefined symbol will thus usually hang up immediately rather than wipe out assorted areas of core before causing recognizable trouble. A branch to an undefined statement will cause a hangup on a 69 op code; a reference to an undefined FORMAT will hang on an address of $0000\neq$. Attempting to perpetrate arithmetic operations on the record mark will generally hang up before any grave problems are generated.

This feature also makes possible some obscure uses of the ASSIGN statement through the judicious placement of intentionally undefined variables; for example, an array may be moved as a single record.

SYSTEM OPTIONS

The option of punching source statements and symbol table has been removed. They may optionally be listed on the typewriter. Whether the listing of the program is being produced or not, the object address of each statement is listed. This list may be used together with an off-line source listing for debugging. The symbol table may be listed with or without the function names, or not listed at all. If the symbol table is not listed, the highest core address below the symbol table will be typed, to show the size of the table.

No compilation error is generated if the program runs into the symbol table. Instead, the subroutine loader keeps track of the end of the program (including function subroutines) and the beginning of the table, and will type

NNNNN DIGITS OVERLAP

if the program overlaps the table, where NNNNN is the amount of overlap. If there is no overlap, the message

NNNNN DIGITS REMAINING

will be typed to indicate the amount of room available for program extension. It is believed that this pair of messages is somewhat more informative than a compilation error (or no error indication if the overlap is due to the function subroutines).

If desired, a map listing showing the entry point for each function subroutine used will also be typed; this saves hunting in the symbol table for the address.

Two sets of Class A subroutines (fixed and free format) are provided as in PDQ. In addition, there are two sets of Class C (relocatable function) subroutines. The second set is a stripped version which does not provide for fixed-point arguments (except in FLT). Use of this deck will permit more core for the object program, but will generate garbage if functions (again excluding FLT) of fixed-point arguments are called for in the object program.

There exists a 1443 version of CHITRAN using essentially the 1443 PDQ patches written, surprisingly, by J. W. Holmes.

A load-and-go version for 40K is being written. Language specifications will be same as 20K CHITRAN; details are presently of questionable relevance (i.e., unavailable).

A derivative of CHITRAN, though not properly a version of it, is SEX FORTRAN, a system which through certain restrictions on input and output format permits the object program to begin in location 03000. The language of SEX includes CHITRAN (except for REREAD) as a subset. In addition, there are the SELECT and EXTERNAL statements which define I/O, and a set of instructions related to ten SENSE LIGHTS. As an added attraction, SEX generates a shorter symbol table than CHITRAN (which is shorter than PDQ, which is...).

SELECT defines the I/O device and the format; EXTERNAL generates the actual I/O instructions. The standard PRINT, PUNCH, etc., are translated by the processor into SELECT and EXTERNAL. The SENSE LIGHTS may be referenced either numerically or symbolically; they may be turned on or off singly or collectively. Examples of statements related to the LIGHTS are:

```
LIGHT  3 OFF
LIGHT  N ON
LIGHTS   ON
IF(SENSE LIGHT Q) 3,4
```

A detailed description of the SENSE LIGHT instruction and of SELECT and EXTERNAL may be found in the proceedings of the May 1964 Eastern Region meeting, in the paper on SEX FORTRAN for paper tape.

GASPS

ULLR

EATS

Peter G. Boekhoff 3193

Fleet Information Systems Group
General American Transportation Corporation
131 South Wabash Avenue, Chicago, Illinois
312-346-4123 Ext. 585

March 3, 1965

GASPS

General American SPS is a paper-tape system written to get away from the inadequacies of existing tape systems (SP-008 and SP-021). GASPS is essentially a rewrite of SP-008; the author found it impossible to live with the card-image output of SP-021 for a program of any great length. One deletion was made from SP-008: GASPS has no subroutines. They were unnecessary for the application (compilers) for which GASPS was written, and the author decided that better use could be made of the core used to process them. GASPS possesses four major advantages over the older systems: processing is faster; the label table is larger; the language has been expanded; printed and/or punched output may be suppressed, e.g. for diagnostic use as pre-assembler.

Processing speed was increased by rewriting the op-code table and the op-code and label-table scans, along with some streamlining of the other routines.

The label table was enlarged, of course, by shortening the processor. This was done by deleting the macro routines and tightening the remainder of the processor. Even with the addition of some new source-language statements and the corresponding assembly routines, the entire processor except for the op-code table occupies less than 10K. This was a secondary objective of GASPS; it makes possible the handling of processor addresses as 4-digit fields, so that GASPS will run in a machine with IA (which, however, GASPS does not use). The label table will handle up to 515 entries.

When GASPS is loaded, it types the message

```
SWITCH 1 ON TO SUPPRESS PUNCH
SWITCH 4 ON TO SUPPRESS PRINT
THEN PRESS START.
```

and halts. When START is pressed, it adjusts itself accordingly and types

```
RESET SWITCHES FOR ASSEMBLY
THEN PRESS START.
```

Assembly begins when START is pressed. This feature can save considerable time when an assembly listing is not needed. GASPS requires just under 3 hours to assemble itself with a full listing; with no listing, the time is less than half an hour.

Declarative mnemonics added to GASPS include SEND (somehow absent from SP-008), which as usual simply halts the processor.

DACF (Define Alpha Constant with Flags) and DACN (Define Alpha Constant with No flag) are identical to DAC except that DACN omits the high-order flag and DACF flags each two-digit character.

DCNF (Define Constant with No Flag) omits the high-order flag of a DC and assigns the label and/or address to the low-order position.

DSCF (Define Special Constant with Flag) generates the high-order flag and assigns the label and/or address to the high-order position.

DNB (Define Numeric Blanks) does not define numeric blanks, but is included so that card programs may be assembled without hanging up. It is assembled as a DS.

DSSA (Define Special Symbolic Address) omits the high-order flags from the generated addresses. The label, if any, is equated to the high-order position of the first address.

Defining a constant (with any of the 4 DC statements) with a preceding minus sign will flag the low-order digit of the constant.

Examples:

DC	2,@	$\bar{0}\neq$
DC	2,-@	$\bar{\bar{0}}\neq$
DC	2,-12	$\bar{1}\bar{2}$
DCNF	2,-12	$1\bar{2}$
DSC	2,-12	$1\bar{2}$
DSC	2,-@	$0\bar{\neq}$

Imperatives added include the missing BI and BNI instructions for I/O and parity indicators (BRC, BWC, BME, BMO, BNRC, BNWC, BNME, BNMO), card commands (including BLC and BNLC), edit

instructions (MF, TNS, TNF), and Model II instructions (BS, TRNM). Index-register-related instructions are not included.

Also added are the mnemonics B7, BB2, and BB7, which are defined by:

```
B7    is equivalent to  B
                        DORG *-4

BB2   is equivalent to  BB
                        DORG *-9
```

Definition of BB7 is left as an exercise for the reader.

Inclusion of the @ symbol in the third (flag) operand of any imperative will cause a record mark to be placed in Q11 of the instruction. The @ comes at the end of the third operand. Examples:

```
TDM   01234,56789,@          15 01234 5678≠

TFM   444,,10@               16 00444 000Ō≠

TFM   444,,@                 16 00444 Ō000≠

TFM   444,,1011@             16 00444 000Ō̄≠

B7    1000,24680,0246810@    4̄9 0̄1̄0̄0̄0̄
```

GASPS was written for use in a 20K machine; it will, however, adjust itself to take advantage of larger capacity. If it is used in a Model II with index registers on, it will turn them off. Thus GASPS may be used in any paper-tape 1620.

Division is not used in address arithmetic; the only valid operators in address operands are +, -, and *. The remaining symbols

.) $ / (= @

may be used in labels. Using one of these special characters as the first character of a label, however, is not recommended.

ULLR

The Unlimited-Length Label Referencer is available for either tape or cards; the two programs are essentially identical. The program occupies just under 3000 positions of core, leaving 17000 (in 20K) for the table. ULLR does not use IA. It will expand its own table if the machine is larger than 20K.

ULLR reads an SPS source program and produces a listing of all labels in alphabetical order, with the line numbers where they are defined and all line numbers where they are used. There is no diagnostic checking. An undefined label will be declared by ULLR to have been defined at line 99999; a multiply-defined label will be listed once for each definition.

When ULLR encounters a DEND statement, or when the table is full, the output is listed on the typewriter, in the usual label-referencer format. 61 lines are typed per page, followed by 5 blank lines. At the end of the output, ULLR halts and reinitializes; another source program (or the remainder of the current one, in case of table overflow) may be processed by poking START.

ULLR will copy all source-program remarks which precede the first SPS statement; the program identification may thus be included in the listing.

ULLR for tape assumes that the input is in GASPS; that is, that all mnemonics are imperatives except DAC, DACF, DACN, DAS, DC, DCNF, DEND, DNB, DORG, DS, DSA, DSB, DSC, DSCF, DSS, DSSA, and SEND. Card ULLR is the same except that SEND is omitted from the list.

Editing and Titling System
(Utility Program)
Program Revision and Tape Titling System
Accepts SPS Source

EATS is a utility program for paper tape. It consists of two parts, either of which may be used independently.

Input to the Editing portion is an SPS source program, with or without line numbers. EATS assigns sequential line numbers to its output, in increments of 10, beginning with the number specified by the user. EATS simply renumbers and copies the source program and clears record marks in the op-code field, except when told otherwise by switch settings; for this normal course of events, switch 4 is turned on and 1 and 2 off (switch 3 tells EATS whether there are line numbers in the input). If switch 2 is turned on, EATS will read the next tape record and halt. When START is pressed, that record is ignored and the next one is read. If switch 4 is turned off, EATS accepts the next statement from the typewriter. Switch 4 may be turned on to correct typing errors in the usual way. If switch 1 is on, EATS reads a tape record and types it, and waits for a typed correction (if the statement is correct, RELEASE and START will copy it to the output program). Thus additions, deletions, and corrections may be made. The editing program, after it punches the DEND statement, drops into the titling program.

The titling program accepts a title from the typewriter and punches reasonable visual facsimiles of the characters, so that the tape identification can be seen on the tape; this is simply an improved version of a similar program already in the library. Thus the usual scheme of things is for EATS to edit an SPS source program and punch its title on the end of the tape.

The first instruction in EATS is a test of switch 1; if this switch is on, control goes directly to the titling routine. Since EATS occupies locations 18100-19999, titles may be punched on the output from a normal 20K FORTRAN or SPS batch run without destroying the language processor. EATS must of course be reloaded each time it is to be used in this way.

THE USE OF MARK SENSING EQUIPMENT IN THE PREPARATION, PROCESSING, AND EXECUTION OF STUDENT WRITTEN PROGRAMS

H. B. KERR
DIRECTOR, COMPUTER CENTER
TENNESSEE TECH
COOKEVILLE, TENNESSEE

ONE OF THE BIGGEST PROBLEMS FACING THE PERSON RESPONSIBLE FOR THE ADMINISTRATION OF A COMPUTER CENTER IN AN EDUCATIONAL INSTITUTION IS THE PREPARATION, PROCESSING AND EXECUTION OF STUDENT WRITTEN PROGRAMS. FOR LO THESE MANY YEARS, HOT ARGUMENTS HAVE BEEN WAGED ON THE RELATIVE MERITS OF AN ...OPEN... VERSUS A ...CLOSED... COMPUTING FACILITY, THAT IS, WHETHER THE ..USER..OF THE COMPUTING FACILITY SHOULD HAVE ...HANDS ON... EXPERIENCE WITH BOTH ON-LINE AND OFF-LINE DATA PROCESSING EQUIPMENT. IT IS NOT THE INTENTION OF THE SPEAKER TO EITHER ADD FUEL TO THIS CONTROVERSEY OR TO TAKE SIDES, ALTHOUGH FROM EXPERIENCE GAINED OVER THE PAST FOUR YEARS, I CAN SEE MERIT IN BOTH SIDES OF THE ARGUMENT. MANY OF THE LARGER COMPUTER CENTERS HAVE FOUND IT EXPEDIENT TO OPERATE ON THE ...CLOSED... BASIS AND HAVE STAFF HIRED ON TO KEY PUNCH THE STUDENT WRITTEN PROGRAMS. OTHER INSTALLATIONS, ONE OF WHICH I ADMINISTER, HAVE FOUND IT NECESSARY TO PROVIDE PERSONNEL TO OPERATE THE COMPUTERS AND CERTAIN ASSOCIATED EQUIPMENT, BUT DO NOT HAVE THE TIME OR PERSONNEL TO PREPARE EACH STUDENT WRITTEN PROGRAM FOR COMPUTER EXECUTION. IT IS, THEREFORE, THE PROCESSING OF STUDENT WRITTEN PROGRAMS PREVIOUS TO COMPUTER COMPILATION AND EXECUTION ABOUT WHICH I WISH TO MAKE A FEW COMMENTS, AND, DURING THE LATTER PART OF MY TALK, MENTION SEVERAL OTHER USES OF MARK SENSE EQUIPMENT.

AT TENNESSEE TECH, WE HAVE TWO TYPES OF STUDENT WRITTEN PROGRAMS TO BE EXECUTED -- MACHINE LANGUAGE PROGRAMS AND FORTRAN PROGRAMS. WE HAVE AN AVERAGE OF 150 STUDENT WRITTEN MACHINE LANGUAGE PROGRAMS AND 175 FORTRAN PROGRAMS TO PROCESS EACH WEEK, IN ADDITION TO THE EXECUTION OF NUMEROUS LIBRARY, TEST SCORING, TEST ASSEMBLY, AND PAYROLL PROGRAMS. THE ONLY PEOPLE DIRECTLY ASSOCIATED WITH THE COMPUTER CENTER, I. E., MYSELF AND MY ASSISTANT, ALSO MUST TEACH FROM FOUR TO EIGHT CONTACT HOURS EACH WEEK. THE BULK OF THE ROUTINE DATA PROCESSING WORK MUST, THEREFORE, BE DONE BY FROM FOUR TO SIX STUDENT ASSISTANTS. THE PERSONNEL IS SIMPLY NOT AVAILABLE TO PREPARE SUTDENT WRITTEN PROGRAMS AND DATA FOR PROCESSING.

TO OVERCOME THE ABOVE MENTIONED PROBLEMS AND TO MAKE THE WORK LOAD OF THE COMPUTER CENTER MORE BEARABLE, WE HAVE GONE TO EXTENSIVE USE OF MARK SENSING EQUIPMENT -- NAMELY, MARK SENSE CARDS, HIGH GRAPHITE PENCILS AND AN IBM 514 REPRODUCING PUNCH EQUIPPED WITH TWENTY-SEVEN POSITIONS OF MARK SENSE BRUSHES. WE HAVE ALSO MADE FREE USE OF CERTAIN IBM 1620 PROGRAMS WHICH LEND THEMSELVES TO ...MARK-SENSE... OPERATION, AS WELL AS FREELY COPYING SUCCESSFUL TECHNIQUES USED BY OTHER SCHOOLS. INCIDENTALLY, I WANT TO MAKE IT CLEAR THAT I AM MAKING NO CLAIM AS TO ORIGINALITY IN THE CREATION OF ANY OF THE SYSTEMS I AM CURRENTLY USING. I SIMPLY WANT TO COMMENT UPON THE APPLICATION OF SOME SPECIAL TECHNIQUES TO EXISTING PROGRAMS AND IDEAS.

FORTRAN CODING CARDS

IN THE SUMMER OF 1964, REPRESENTATIVES OF THE U. S. MILITARY ACADEMY AT WEST POINT PRESENTED A PAPER BEFORE THE AMERICAN SOCIETY FOR ENGINEERING EDUCATION ON THE PREPARATION OF FORTRAN PROGRAMS BY USE OF SPECIAL MARK SENSE CARDS. THIS IDEA LOOKED PROMISING TO THOSE OF US AT TENNESSEE TECH AND, WITH THE KNOWLEDGE AND CONSENT OF THE ORIGINATORS

OF THE WEST POINT CARDS, WE HAD MARK SENSE CARDS PREPARED FOR FORTRAN II D, WROTE A PROGRAM TO ...DECODE... THIS INFORMATION AND PUNCH FORTRAN STATEMENT CARDS FOR PROCESSING IN THE USUAL FASHION. A LATER MODIFICATION WAS TO INCORPORATE THE ...DECODER... PROGRAM AS A PROGRAM ON THE 1311 DISK PACK, UNDER MONITOR CONTROL, TO ELIMINATE THE PUNCHING OF THE FORTRAN SOURCE PROGRAM, GOING DIRECTLY TO THE COMPILATION AND EXECUTION PHASES. AT THE TIME OF THIS WRITING, THE LATTER SYSTEM IS ONLY PARTLY COMPLETE.

THE STUDENT PREPARES HIS FORTRAN PROGRAM BY MARKING THE APPROPRIATE SLOT IN THE FORTRAN STATEMENT CODING CARD. CERTAIN CONTROL WORDS SUCH AS IF, DIMENSION, AND IF SENSE SWITCH MAY BE FORMULATED BY BY MARKING ONLY ONE SLOT. ARITHMETIC STATEMENTS, STATEMENT NUMBERS, ETC. MAY ALSO BE FORMULATED, BY MARKING SINGLE OR MULTIPLE SLOTS IN A COLUMN . THE PROGRAMMER MAY CONTINUE A STATEMENT FROM ONE CARD TO THE NEXT BY MARKING A SLOT PROVIDED FOR THAT PURPOSE. CONTINUATION FORTRAN STATEMENT CARDS ARE PUNCHED AS NEEDED UP TO A MAXIMUM OF FOUR, AS PERMITTED IN FORTRAN II D. THE FORTRAN ...DECODER... PROGRAM IS AVAILABLE FROM THE LIBRARY (PROGRAM NO. 1.3.015). UNTIL THE ACQUISITION OF ITS NEW 40K, 1620 - 1311 SYSTEM, TENNESSEE TECH DECODED, COMPILED AND EXECUTED THE ...MARK SENSED... FORTRAN PROGRAMS ON A 20K 1620 COMPUTER, USUALLY COMPILING THE PROGRAMS USING THE PDQ PROCESSOR (NO CONTINUATION CARDS BEING PERMITTED, OF COURSE). WE HAVE BEEN IN OPERATION UNDER MONITOR FOR ONLY A SHORT TIME. HOWEVER, IT APPEARS THAT A VERY REAL TIME SAVING WILL BE EFFECTED BY BATCH COMPILING IN ONE STEP ON THE 1620 - 1311 SYSTEM. I AM SORRY TO SAY THAT THAT I CAN QUOTE NO AVERAGE COMPILATION TIMES FOR THE 1620 - 1311 SYSTEM. WHEN EXECUTING THE DECODER PROGRAM SEPARATELY, THE FORTRAN SOURCE PROGRAM IS PRODUCED AT APPROXIMATELY PUNCH SPEED, I. E., 250 CARDS PER MINUTE, IN OUR CASE. IF ANYONE WOULD BE INTERESTED IN TRYING OUT THIS SYSTEM, TENNESSEE TECH WOULD BE GLAD TO GIVE PERMISSION TO USE OUR ELECTRO PLATES. THE DECODER PROGRAM, AS WE HAVE SAID BEFORE, IS AVAILABLE FROM THE LIBRARY.

MACHINE LANGUAGE PROGRAMMING

IN ADDITION TO STUDENT WRITTEN FORTRAN PROGRAMS, WE ALSO HAVE AT LEAST 125 TO 150 STUDENT WRITTEN MACHINE LANGUAGE PROGRAMS TO EXECUTE EACH WEEK. TO FACILITATE THE HANDLING OF THESE PROGRAMS, TENNESSEE TECH IS MAKING USE OF A SERIES OF PROGRAMS CALLED ...MARCAT..., WRITTEN BY PROFESSOR GUY RICKER OF NEW JERSEY CITY STATE COLLEGE. MANY OF YOU MAY HAVE HEARD HIS PRESENTATION AT THE FALL 1963 1620 USERS GROUP CONVENTION IN PITTSBURGH. PROFESSOR RICKER VERY KINDLY MADE HIS PROGRAMS AND WRITE UPS AVAILABLE TO ME AND I HAVE, WITH CERTAIN MODIFICATIONS AND RESTRICTIONS, INSTITUTED THEIR USE IN MACHINE LANGUAGE PROGRAMMING COURSES. OF COURSE, AS SOME OF YOU MAY KNOW, THIS PROGRAM MAY ALSO BE USED IN SPS PROGRAMMING.

FOR THE BENEFIT OF THOSE WHO ARE NOT FAMILIAR WITH ...MARCAT..., IT IS A PROGRAM WHICH WILL EXECUTE A STUDENT WRITTEN PROGRAM UNDER CONTROLLED CONDITIONS, CHECK THE ANSWER OR ANSWERS, AND (WITH MARCAT II) INITIATE AND PUNCH OUT A TRACE OF THE STUDENT*S PROGRAM IF THE ANSWERS FAIL TO CHECK OUT. IN ORDER TO MAKE THIS SYSTEM APPLICABLE TO MARK SENSE CARD INPUT AND BATCH PROCESSING, WE HAVE MADE CERTAIN RESTRICTIONS ON THE WRITING OF STUDENT PROGRAMS, SUCH AS -- THE ELIMINATION OF ...CONSTANT... CARDS, AS PROVIDED FOR IN PROFESSOR RICKER*S PROGRAM, EVERY STUDENT*S PROGRAM MUST ORIGINATE AT 00500, THE INSTRUCTIONS MUST BE CONSECUTIVE AND DATA INPUT CAN BE NO LONGER THAN 27 DIGITS IN LENGTH.

IN OPERATION, EACH CLASS IS ASSIGNED A UNIQUE ONE DIGIT NUMBER

AND EACH STUDENT A UNIQUE TWO DIGIT NUMBER WHICH ALONG WITH THE PROBLEM NUMBER IS MARK SENSED INTO THE I. D. CARD TO IDENTIFY THE STUDENT, COURSE, AND PROBLEM NUMBER. AT THE BEGINNING OF THE COURSE, EACH STUDENT IS GIVEN A PRE-PUNCHED PACKET OF CARDS AS FOLLOWS

10 I. D. CARDS (MARK SENSE)
50 PROGRAM CARDS (MARK SENSE)
1 TRAILER CARD
10 DATA CARDS (MARK SENSE)

IN ADDITION, EACH STUDENT IS GIVEN A SET OF INSTRUCTIONS AND AN EXPLANATION OF HOW TO INTERPRET THE ...MARCAT... OUTPUT. PROBLEM STATEMENT SHEETS ARE DISTRIBUTED BY THE INSTRUCTOR. THE PROBLEM STATEMENT MUST, OF COURSE, BE WORDED VERY CAREFULLY IN ORDER THAT EACH STUDENT UNDERSTAND EXACTLY WHERE HIS DATA AND HIS ANSWERS ARE TO BE STORED, AS WELL AS A THOROUGH STATEMENT OF THE PROBLEM. THE PROPER PLACEMENT OF THE DATA AND ANSWER IS IMPORTANT BECAUSE, WHEN EXECUTING UNDER ...MARCAT... CONTROL, THE ...MARCAT... PROGRAM WILL NOP ALL INPUT-OUTPUT INSTRUCTIONS, USING THE CORRECT DATA AS THAT STORED IN UPPER MEMORY.

ONE OF OUR SHARPER STUDENT ASSISTANTS AT TENNESSEE TECH HAS WRITTEN A SIZEABLE PROGRAM WHICH, WHEN MARRIED TO THE EXISTING MARCAT II, READS IN THE STUDENT WRITTEN I. D. CARD, PROGRAM CARDS, TRAILER CARD AND DATA CARD(S), INITIATES A ...FREE RUN... OF THE PROGRAM, AND THEN BRANCHES (BY MEANS OF PATCHES) INTO THE MARCAT PROGRAM FOR AN ANALYSIS, EVALUATION AND (IF NECESSARY) TRACE OF THE STUDENT WRITTEN PROGRAM. THIS PROGRAM ALSO PUNCHES OUT CARDS WHICH IDENTIFY OUTPUT TO FOLLOW AND ALSO PROVIDE SEPARATION BETWEEN STUDENT PROGRAMS. IMPLEMENTATION OF THIS PROGRAM HAS DECREASED THE NECESSARY PROCESSING TIME TO ABOUT 10 PERCENT OF THAT REQUIRED BY EARLIER SYSTEMS EMPLOYED AT TENNESSEE TECH. 80 - 80 LISTINGS ARE MADE OF ALL STUDENT PROGRAMS, OUTPUT CARDS, PROGRAM EVALUATION AND TRACE (IF ONE WAS PUNCHED) AND RETURNED TO THE STUDENT.

EVERY ATTEMPT IS MADE TO PROCESS THE STUDENT WRITTEN PROGRAMS WITHIN A TWO OR THREE HOUR PERIOD. MANY STUDENTS, WHEN THEIR PROGRAMS DEVELOP BUGS UPON EXECUTION, RETURN THE SAME PROGRAM SEVERAL TIMES FOR PROCESSING. AS A MATTER OF FACT, THEY ARE ENCOURAGED TO DO SO , AND PROPER NOTE IS MADE OF THEIR REPEATED EFFORTS. THE INSTALLATION OF THIS SYSTEM HAS APPEARED TO GIVE THE STUDENT PROGRAMMERS CONSIDERABLE STIMULI TO LEARN COMPUTER PROGRAMMING. THEY ARE ENCOURAGED TO ...HANG AROUND... THE COMPUTER CENTER AND OBSERVE THE PROCESSING TECHNIQUES. IN FACT, WHEN ONE OF THE COMPUTERS IS AVAILABLE, STUDENTS WITH THE INTEREST AND TIME TO DO SO HAVE BEEN PERMITTED TO LEARN CONSOLE OPERATION, WITH ASSISTANCE FROM THE STUDENT ASSISTANTS, TIME PERMITTING AND, WHEN THEIR COMPETENCE IS PROVED, THEY ARE ISSUED LICENSES TO OPERATE THE VARIOUS COMPUTERS AND PERIPHERAL EQUIPMENT.

THERE HAVE BEEN SEVERAL BUGS AND INCONSISTENCIES WHICH HAVE DEVELOPED IN THE OPERATION USING MARCAT CONTROL, BUT, ALL IN ALL, THE RESULTS HAVE BEEN QUITE SATISFACTORY.

TEST ASSEMBLY

SEVERAL INSTRUCTORS AT TENNESSEE TECH ARE MAKING USE OF AN ...EXAMINATION ASSEMBLY... SERVICE OFFERRED (THE OFFERING BEING SOMEWHAT RESTRICTED DUE TO THE LACK OF AVAILABILITY OF KEY PUNCH PERSONNEL). THIS PROGRAM IS AVAILABLE FROM THE LIBRARY AS NO. 13.0.012. USE OF THIS PROGRAM NECESSITATES THAT A POOL OF EXAMINATION QUESTIONS BE PUNCHED INTO CARDS. PREVIOUS TO THE EXECUTION OF THE PROGRAM,

SELECTED QUESTIONS ARE CALLED FOR (THE INSTRUCTOR USUALLY MAKING HIS CHOICE BY MARK SENSE CARD), THE COMPUTER PROGRAM SEARCHES THE POOL OF QUESTIONS FOR THE SELECTED QUESTIONS, ASSIGNS A NEW SEQUENCE NUMBER AND PUNCHES THE RE-SEQUENCED QUESTIONS INTO CARDS. AN IDENTIFICATION CARD, INSTRUCTION CARDS, ETC., ARE USUALLY PUT ON THE FRONT OF THE DECK AND A DITTO MASTER CUT ON THE 407 ACCOUNTING MACHINE (USING AN 80 - 80 PANEL). THE QUESTIONS MAY BE OF VARIABLE LENGTH AND MAY BE OF THE OBJECTIVE TYPE OR ESSAY TYPE. WIDE USE IS MADE OF THIS PROGRAM IN THE COMPUTER COURSES WHERE FORTRAN AND MACHINE LANGUAGE IS TAUGHT.

TEST SCORING

AT TENNESSEE TECH, WE MAKE CONSIDERABLE USE OF THE NORTHEASTERN TEST SCORING PROGRAM (LIBRARY NO. 13.0.003). THIS IS A COMPUTER PROGRAM WHICH WILL PERFORM THE OPERATION OF SCORING AN OBJECTIVE TYPE EXAMINATION CONSISTING OF UP TO 150, 5 CHOICE QUESTIONS, PERFORM A STATISTICAL ANALYSIS OF THE RESULTS, AND YIELD AN ITEM ANALYSIS, QUESTION BY QUESTION.

THIS PROGRAM HAS ACHIEVED A WIDE USAGE THROUGHOUT THE COLLEGE, AND HAS BEEN INSTRUMENTAL IN CAUSING A SAVING IN INSTRUCTION TIME, PERMITTING LARGER CLASSES TO BE HANDLED IN CERTAIN COURSES, WITHOUT IMPOSING AN UNDUE HARDSHIP UPON THE INSTRUCTOR. IN MANY COURSES, THE OUTPUT FROM THE NORTHEASTERN TEST SCORING PROGRAM IS PRESERVED AND AT THE END OF THE QUARTER, IS UTILIZED IN CONJUNCTION WITH THE GRADE AVERAGING PROGRAM (DESCRIBED LATER) IN DEVELOPING A COURSE GRADE.

GRADE AVERAGING AND REPORTING

AT TENNESSEE TECH, MARK SENSE CARDS ARE USED IN THE PREPARATION OF GRADE REPORTS. THE MILITARY SCIENCE DEPARTMENT REPORTS EXAM GRADES, MERITS AND DEMERITS, DRILL GRADES, ETC., COMPLETELY ON MARK SENSE CARDS. THE COMPUTER CENTER HAS WRITTEN A SPECIAL PROGRAM WHICH CALCULATES COURSE GRADES, MILITARY STANDING, DEMERITS, ETC. AND PREPARES THE REPORT FOR THE ADMISSIONS AND RECORDS OFFICE.

A MORE GENERAL COMPUTER PROGRAM HAS BEEN WRITTEN WHICH UTILIZES THE OUTPUT OF THE ...NORTHEASTERN TEST SCORING PROGRAM... IN CALCULATING AND REPORTING COURSE GRADES. THIS PROGRAM PERMITS ANY NUMBER OF HOURLY TESTS AND ONE FINAL EXAMINATION TO BE WEIGHTED IN ACCORDANCE WITH THE WISHES OF THE INSTRUCTOR.

DATA COLLECTION USES

LIMITED USE IS MADE OF MARK SENSE CARDS IN THE COLLECTION OF DATA TO BE USED IN MAKING CERTAIN STATISTICAL ANALYSES. MANY TIMES, CERTAIN OF THE FACULTY DESIRE THE CORRELATION AND ANALYSIS OF CERTAIN DATA, BUT HAVE NEITHER THE TIME OR THE INCLINATION TO COME TO THE COMPUTER CENTER TO PREPARE THE DATA FOR INTRODUCTION TO A COMPUTER PROGRAM. IN THIS EVENT, THE FACULTY MEMBER IS GIVEN MARK SENSE CARDS, AND PENCILS AND INSTRUCTED AS TO THEIR USE, AFTER WHICH HE PERFORMS THE MARKING OF THE CARDS AT HIS CONVENIENCE IN HIS OFFICE. THE MARKED CARDS ARE RETURNED TO THE COMPUTER CENTER FOR PROCESSING.

THE FACT THAT THERE ARE ONLY 27 MARK SENSE POSITIONS AVAILABLE PER SIDE OF A CARD DOES, OF COURSE, IMPOSES SOME RESTRICTIONS AND, IN SOME CASES CALLS FOR A SPECIAL PROGRAM TO COMBINE DATA CARDS, ETC.

REGISTRATION SECTIONING

AT TENNESSEE TECH, REGISTRATION SECTIONING IS DONE BY COMPUTER

(USING A RE-WRITTEN VERSION OF THE IBM ...STUDENT... PROGRAM, LIBRARY NO. 10.3.017). IN ORDER TO FACILITATE THE COLLECTION OF INFORMATION REGARDING THE COURSES DESIRED BY THE VARIOUS STUDENTS, EXTENSIVE USE OF MARK SENSE CARDS HAS BEEN MADE. CONSIDERABLE DIFFICULTY HAS BEEN ENCOUNTERED IN THIS APPLICATION, DUE TO THE HUMAN FACTOR, I. E., THE MISTAKES MADE BY THE STUDENTS IN MARKING THEIR COURSE REQUEST CARDS. THE DIFFICULTY HERE IS NOT IN THE MARKING OF THE CARDS AS MUCH AS IT IS THE INABILITY OF THE STUDENT TO READ HIS COURSE LIST AND SELECT THE RIGHT COURSES. WHEN HIS COURSE REQUESTS ARE PROCESSED BY THE COMPUTER AND HE FINDS HIMSELF ASSIGNED TO AN ENGLISH COURSE (WHICH HE REQUESTED) INSTEAD OF A HISTORY COURSE (WHICH IS WHAT HE INTENDED TO SELECT), THE STUDENT INVARIABLY BLAMES THE COMPUTER FOR THE ERROR. THE STUDENT NEWSPAPER ALSO LOVES TO DWELL UPON THE CASE OF THE HAPLESS GIRL WHO IS ASSIGNED BY THE ...ELECTRONIC BRAIN... TO A SECTION OF MILITARY SCIENCE.

HOUSING RECORDS

THE DEAN OF STUDENTS OFFICE AT TENNESSEE TECH IS REQUIRED TO SUBMIT A NUMBER OF REPORTS TO THE STATE BOARD OF EDUCATION REGARDING THE HOUSING INFORMATION ON ALL STUDENTS. THESE REPORTS, AS WELL AS GENERAL HOUSING RECORDS, ARE NOW DONE BY THE USE OF SPECIAL MARK SENSE CARDS, AND THE MARK SENSE HOUSING CARD IS INCLUDED IN THE REGISTRATION PACKET FOR EACH STUDENT. AS AN ADJUNCT TO THIS PROCESS AUTOMOBILE REGISTRATION NUMBERS ARE MARKED ON THE HOUSING CARD AND ADDITIONAL LISTINGS MADE FOR THE DEAN OF STUDENT*S OFFICE, WITH THE HOUSING CARDS ALPHABETIZED BY STUDENT NUMBER AND ALSO IN NUMERICAL ORDER BY AUTOMOBILE REGISTRATION NUMBER. A CONSIDERABLE SAVINGS IN CLERICAL WORK HAS BEEN EFFECTED BY USE OF THE HOUSING CARDS AS WELL AS AN ACCELERATION OF THE PREPARATION OF THE REPORTS.

STUDENT ELECTION RETURNS

FOR THE PAST YEAR, THE ASSOCIATED STUDENT BODY AT TENNESSEE TECH HAS RUN THEIR ELECTIONS USING MARK SENSE CARDS AS BALLOTS. AFTER PROCESSING THROUGH THE 514 REPRODUCING PUNCH, THE BALLOTS WERE TABULATED USING THE LIBRARY PROGRAM ...CAST... (LIBRARY NO. 6.0.146). THE STUDENT OFFICIALS FEEL THAT THIS SYSTEM YIELDS MUCH QUICKER AND MORE ACCURATE RESULTS, AND AT THE SAME TIME DECREASES THE POSSIBILITY OF TAMPERING WITH THE BALLOTS.

SUMMARY

THE USE OF MARK SENSE EQUIPMENT AT TENNESSEE TECH HAS GROWN SO RAPIDLY THAT THE PROPER OPERATION OF THE 514 REPRODUCING PUNCH HAS BECOME EVEN MORE IMPORTANT THAN THAT OF ONE OF THE COMPUTERS. IN THIS RESPECT WE ARE BLESSED BY HAVING A SPARE 514 WITH MARK SENSE BRUSHES IN ANOTHER OFFICE OF THE CAMPUS. WE FEEL THAT THE USE MADE OF MARK SENSE EQUIPMENT IS, TO US, AN ABSOLUTE NECESSITY RATHER THAN A CONVENIENT LUXURY. THE PROBABILITY OF FUTURE ACQUISITION OF MORE UP-TO-DATE OPTICAL MARK SENSE EQUIPMENT WILL UNDOUBTEDLY STILL FURTHER ACCELERATE THE USAGE OF THIS METHOD OF INFORMATION COLLECTION.

USE OF FORTRAN MARK SENSE CARDS

DESCRIPTION - THIS PROGRAM WAS DEVELOPED TO OPERATE IN CONJUNCTION WITH MARK SENSE CARDS OF A FORMAT SIMILAR TO THE ... CADETRAN ... FORTRA MARK SENSE CARDS DEVELOPED BY THE U. S. MILITARY ACADEMY. IT WAS DECIDED TO MODIFY THE CARD FORMAT OF CADETRAN TO MAKE IT SUITABLE F USE WITH FORTRAN II D FOR THE IBM 1311 DISK FILE.

BY MARKING THE SPECIAL CARDS WITH A PENCIL CONTAINING A HIGH GRAPHI CONTENT (SEE APPENDIX), IT IS POSSIBLE FOR A BEGINNING (OR EXPERIENCED) FORTRAN PROGRAMMER TO PREPARE HIS FORTRAN STATEMENT CARDS INDEPENDENT OF A KEY PUNCH. THE PROGRAMMER SIMPLY FORMULATES HIS FORTRAN STATEMENT CARDS BY MARKING THE APPROPRIATE SLOTS ON THE SPECIALLY PRINTED CARDS. PROVISION IS MADE FOR CONTINUING THE STATMENT FROM ONE MARK SENSE CARD TO THE NEXT AND FOR DENOTING AND PUNCING CONTINUATION CARDS. THE MARKED CARDS ARE THEN PASSED THROUGH A REPRODUCING PUNCH EQUIPPED WITH 27 POSITIONS OF MARK SENSE BRUSHE THE CARDS ARE THEN DECODED BY THE SUBJECT PROGRAM, INTERPRETED (IF DESIRED), AND THEN PROCESSED AS WITH ANY OTHER FORTRAN PROGRAM. A THREE DIGIT SEQUENCE NUMBER IS PLACED IN CARD COLUMNS 78 THROUGH 80 OF THE FORTRAN STATEMENT CARDS.

MARKING THE FORTRAN CARDS -

1. STATEMENT NUMBER - IF IT IS DESIRED TO USE A STATEMENT NUMBER, MARK THE INDICATED SLOTS (USING A FIRM PRESSURE WITH A SHARP POINTED, HIGH GRAPHITE CONTENT PENCIL). ONLY A TWO DIGIT STATMENT IS PERMITTED (WHICH SHOULD BE SUFFICIENT FOR MOST PROGRAM

2. CONTINUATION - IF THE CARD IS A CONTINUATION OF A PREVIOUS CAR THE ... CONTINUATION ... SLOT SHOULD BE MARKED. IF IT IS NOT MARKED, THE SUBJECT PROGRAM ASSUMES THAT THIS IS A NEW STATEMENT.

3. COMMENTS - MARKING OF THE ... COMMENTS ... SLOT CALLS FOR A LETTER C TO BE PLACED IN CARD COLUMN 1, THEREBY MAKING THE FORTRAN STATEMENT A COMMENTS STATEMENT.

4. I/O, FORMATS, CALL, DO, ETC. - MARKING ANY ONE OF THESE SLOTS CALLS FOR THE APPROPRIATE WORD OR WORDS TO BE PLACED IN THE OUTPUT CARD. A BLANK USUALLY FOLLOWS THE WORD OR WORDS. ONLY ONE OF THE SLOTS IN THE THREE SPECIAL COMMAND COLUMNS MAY BE MARKED. FAILURE TO MARK ONE OF THESE SLOTS WILL CALL FOR NO CHARACTERS OR BLANKS TO BE TRANSFERRED TO THE OUTPUT FORTRAN CARD.

5. FIELDS - IN THE TWO MARK SENSE COLUMNS OF EACH FIELD ARE ALL NUMBERS AND CHARACTERS ORDINARILY USED IN FORTRAN STATEMENTS.

   LEFT HAND FIELD - ALL PUNCTUATION ORDINARILY USED IN FORTRAN A MOST OPERATORS, AS WELL AS CERTAIN FREQUENTLY USED SUBROUTINES ARE AVAILABLE IN THESE COLUMNS. IF NO MARK IS MADE IN THESE COLUMNS, NO CHARACTERS, PUNCTUATION OR BLAN ARE TRANSFERRED TO THE OUTPUT FORTRAN CARD.

   RIGHT HAND FIELD - ALL NUMBERS, ALPHABETICAL CHARACTERS (AS WELL AS THE SLASH, INDICATING DIVISION) ARE AVAILABLE IN THESE COLUMNS. IF NO SLOTS ARE MARKED, A BLANK WILL BE PLACED INTO THE OUTPUT FORTRAN CARD. IF A BLANK IS NOT DESIRED, THE 12 ZONE PUNCH ONLY (PRINTED A THROUGH I ON

THE CARD) SHOULD BE MARKED. IF THERE IS NO MORE INFORMATION TO BE PLACED ON THE CARD, ONLY THE 11 ZONE PUNCH (PRINTED J THROUGH R ON THE CARD) SHOULD BE MARKED. MARK THIS SLOT ALONE WILL CAUSE THE SUBJECT PROGRAM TO IGNORE REMAINDER OF THE CARD.

OUTPUT - THE OUTPUT CARDS WILL BE IN THE PROPER FORTRAN FORMAT AS DESCRIED IN THE IBM FORTRAN II D LITERATURE, WITH THE EXCEPTION THAT ONLY TWO DIGIT STATEMENT NUMBER IS PERMITTED. IF DESIRED, THE OUTPUT CARDS MAY BE INTERPRETED FOR EASE IN DEBUGGING.

MARCAT PROGRAM CHECKER - GENERAL DESCRIPTION

THE MARCAT PROGRAM CHECKER IS A PROGRAM THAT HAS BEEN DEVELOPED FOR THE PURPOSE OF CHECKING A STUDENT FORMULATED COMPUTER PROGRAM FOR ERRORS IN OPERATION AND ALSO TO CHECK FOR A VALID ANSWER. THE PROGRAM WAS DEVEL ED BY PROFESSOR GUY W. RICKER OF JERSEY CITY STATE COLLEGE, JERSEY CITY, NEW JERSEY. THE COMPUTER PROGRAM WHICH CHECKS ERRORS IN THE STUDENT PROGRAM OCCUPIES THE TOP 12,000 DIGITS OF MEMORY (LOCATIONS 08,000 THROUC 19,999). FROM 3,500 TO 8,000 IS A STORAGE AREA USED BY THE PROGRAM. THIS MEANS THAT THE STUDENT MUST NOT USED OR ATTEMPT TO USE ANY LOCATION ABOVE 3,500. IN FACT, WITH THE EXCEPTION OF THE LAST INSTRUCTION IN THE PROGRAM AS INDICATED BELOW, THE STUDENT IS PROHIBITED FROM MAKING REFERENCE TO LOCATIONS ABOVE 3,500. TO FACILITATE STUDENT PROGRAMMING EXTENSIVE USE OF MARK SENSE CARDS HAS BEEN MADE, THUS ELIMINATING ANY REQUIRED KNOWLEDGE OF KEY PUNCH OPERATION BY THE STUDENT. IN ORDER TO MAKE PROGRAMMING EASIER FOR THE STUDENT, LOCATION 00500 HAS BEEN ESTAB- LISHED AS THE REQUIRED BEGINNING LOCATION FOR THE STUDENT PROGRAM. HIS PROGRAM MUST BEGIN AT THIS LOCATION AND CONTINUE THROUGH CONSECUTIVE LOCATIONS. ANY CONSTANTS TO BE SUPPLIED TO THE STUDENT PROGRAM MUST BE SUPPLIED BY THE USE OF DATA CARDS (TO BE READ IN BY THE STUDENT PROGRAM). THE LAST INSTRUCTION IN THE STUDENT PROGRAM MUST BE A BRANCH TO THE BEGINNING OF MARCAT ... 49 12000 00000 ... WHICH WILL CAUSE THE TRACING AND CHECKING PROCEDURE TO BEGIN. THERE IS A RECORD MARK AVAILABLE IN LOCATION 00400 THAT THE STUDENT MAY USE. MARCAT DOES NOT PERMIT THE STUDENT TO USE A RECORD MARK AS A PORTION OF ANY INSTRUCTION.

USING MARCAT, THERE ARE TWO WAYS OF OPERATING A STUDENT WRITTEN PRO- GRAM ... RUNNING THE PROGRAM DIRECTLY OR RUNNING UNDER MARCAT CONTROL.

DIRECT RUNNING ... THE PRE-PUNCHED DECK OF CARDS YOU ARE GIVEN, AFTE BEING PROPERLY MARK SENSED AND PUNCHED, CAN BE INCORPORATE WITH OTHER STANDARD CARDS PRESENT IN THE COMPUTER CENTER AND THE PROGRAM RUN AS ANY OTHER PROGRAM. ALL PROGRAMS WRITTEN BY THE STUDENT WILL BE RUN IN THIS FASHION BEFORE RUNNING UNDER MARCAT CONTROL. THE REASON FOR THIS IS TO ELIMINATE THE POSSIBILITY OF THE STUDENT WRITTEN PRO- GRAM DESTROYING THE MARCAT PROGRAM. THE STUDENT IS FREE TO CALL FOR ANY OUTPUT (PREFERABLY TYPEWRITER, IF THE OUTPUT IS NOT TOO LONG) THAT HE DESIRES. THE OUTPUT WILL INCLUDED IN THE PACKET RETURNED TO THE STUDENT BY THE COMPUTER CENTER.

RUNNING UNDER MARCAT CONTROL ... UNDER THE CONTROL OF THE MARCAT PROGRAM, THE STUDENT WRITTEN PROGRAM IS EXECUTED AND THE VALUES OF DATA AND ANSWERS ARE COMPARED AGAINST THOSE STOR IN THE MEMORY OF MARCAT. NOTE IS MADE ON THE CORRECTNESS OF THE STUDENT ANSWERS AND, IF THE ANSWERS ARE INCORRECT, A TRACE IS INITIATED WHICH DOCUMENTS THE RESULT OF EACH INSTRUCTION EXECUTED. THIS TRACE DOCUMENTATION WILL BE LISTED AND RETURNED TO THE STUDENT TO ASSIST HIM IN DEBUGGING HIS PROGRAM. THE INSTRUCTOR WILL BASE HIS GRADE UPON THE RESULTS INDICATED BY THE MARCAT PROGRAM.

STUDENT INSTRUCTIONS FOR MARKING CARDS FOR THE USE OF THE MARCAT PROGRAM CHECKER

THE STUDENT WILL BE GIVEN THREE TYPES OF MARK SENSE CARDS ... IDEN- TIFICATION CARDS, PROGRAM CARDS, AND DATA CARDS. CERTAIN STANDARD INFORMATION HAS BEEN PREPUNCHED INTO THE IDENTIFICATION CARDS AND THE

PROGRAM CARDS. THE CARDS ARE TO BE MARKED AS FOLLOWS (ALL COLUMNS REFERR TO ARE MARK SENSE COLUMNS, NOT CARD COLUMNS).

IDENTIFICATION CARDS (GREEN CARDS) ...

COLUMN 3 ... CLASS NUMBER (TO BE ASSIGNED BY THE INSTRUCTOR)
COLUMNS 4 AND 5 ... STUDENT NUMBER
COLUMNS 6 AND 7 ... PROBLEM NUMBER (TO BE ASSIGNED BY THE INSTRUCTOR).

PROGRAM CARDS (NATURAL CARDS) ...

COLUMN 3 ... CLASS NUMBER (OPTIONAL)
COLUMNS 4 AND 5 ... STUDENT NUMBER (OPTIONAL)
COLUMNS 6 AND 7 ... PROBLEM NUMBER (OPTIONAL)
COLUMNS 11 THROUGH 22 ... THE INSTRUCTION DESIRED. ALL FLAGGE NUMBERS MAY BE INDICATED BY MARKING THE 11 ZONE SLOT AS WELL AS THE NUMBER.

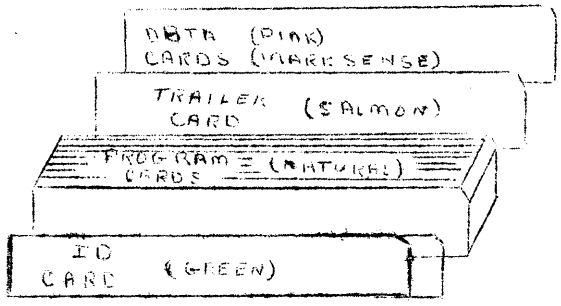
DATA CARDS (PINK CARDS) ...

BEGINNING IN COLUMN 1, THE DESIRED DATA (UP TO A MAXIMUM OF 27 DIGITS) IS TO BE MARKED. FLAGS MAY BE INDICATED BY MARKING TH 11 ZONE PUNCH AS WELL AS THE NUMBER DESIRED. DO NOT SPECIFY RECORD MARKS ON DATA CARDS. THERE IS A RECORD MARK AT LOCATION 00400 AVAILABLE FOR YOU TO USE AT ANY TIME.

WHEN THE CARDS HAVE BEEN PROPERLY MARKED, YOU SHOULD BRING THE CARDS TO THE COMPUTER CENTER, FILL OUT A PROCESSING REQUEST CARD, AND DEPOSIT YOUR DECK AND THE REQUEST CARD IN THE BOX MARKED .... IN ..... THE COMPUTER CENTER PERSONNEL WILL RUN YOUR PROGRAM DIRECTLY AND THEN AGAIN UNDER MARCAT CONTROL. THE RESULTS WILL BE ATTACHED TO YOUR CARD DECK AND LEFT FOR YOU TO PICK UP IN THE BOX MARKED ..... OUT.....

IN ADDITION TO THE MARK SENSE CARDS DESCRIBED ABOVE, THE STUDENT WILL ALSO BE GIVEN A SALMON CARD PRE PUNCHED WITH THE FOLLOWING INFORMATION PRINTED AT THE TOP. (FOR MARCAT PLACE THIS CARD AFTER YOUR LAST PROGRAM CARD - FOLLOW WITH DATA)

THE AFOREMENTIONED CARDS WILL BE ARRANGED AS SHOWN BELOW.



WHEN THE DECK HAS BEEN MARK SENSED AND ASSEMBLED AS SHOWN IN THE SKETCH ABOVE, IT SHOULD BE BROUGHT TO THE COMPUTER CENTER, A PROCESSING REQUEST CARD MADE OUT AND PUT ON TOP OF THE DECK, AND THE COMBINED DECK C CARDS PLACED IN THE BOX MARKED ... IN.

TYPICAL STUDENT PROGRAM PARTIAL ERROR WITH TRACE ON ERROR

```
2304Z0102311198000047380000200100491198O    360000000500490000004900012R00(
2304    360432100500Z                                          J1-     -      R
2304    320432100000Z                                          J1-     -      R
2304    320432400000Z                                          J1-     -      R
2304    320433100000Z                                          J1-     -      R
2304    320433600000Z                                          J1-     -      R
2304    230432304330Z                                          J1-     -      R
2304    210009904335Z                                          J1-     -      R
2304    260435000099Z                                          J1-     -      R
2304    380432100400Z                                          J1-     -      R
2304    491200000000Z                                          J1-     -      R
2304    320433000000Z                                          J1-     -      R
2304    230432304330Z                                          J1-     -      R
2304    220009904340Z                                          J1-     -      R
2304    260435000099Z                                          J1-     -      R
2304    380432100400Z                                          J1-     -      R
2304    491200000000Z                                          J1-     -      R
2304    230433504340Z                                          J1-     -      R
2304    260435000099Z                                          J1-     -      R
2304    380432100400Z                                          J1-     -      R
2304    491200000000Z                                          J1-     -      R
 MARCAT - PLACE THIS CARD AFTER YOUR LAST PROGRAM CARDFOLLOW WITH DATA R
)0001230000601111
)05555N05321J2369
)00000000666N4321
       MARCAT II D OUTPUT
)0001230000601111000000375000000000
)05555N05321J2369000555017R000000000 0
)00000000666N432100000006660000000000

3 04  13 OK  23 NO
)0500360432100500
)0512320432100000                J                                      J
)0524320432400000                0                                      -
)0536320433100000                0                                      -
)0548320433600000                J                                      J
)0560230432304330              J00          005555N            0005555500
)0572210009904335      0005555500            05321            000555017R
)0584260435000099      000555017R      000555017R             000555017R
)0596380432100400
)0608491200000000
3 04  13 OK  23 NO  33 NO
0500360432100500
)0512320432100000                K                                      K
)0524320432400000                0                                      -
)0536320433100000                0                                      -
)0548320433600000                N                                      N
)0560230432304330              K00          0000               -         -
)0572210009904335      0000000000             - 666            0000000666
)0584260435000099      0000000666      0000000666             0000000666
)0596380432100400
)0608491200000000

3 04  13 OK  23 NO  33 NO
```

TYPICAL STUDENT PROGRAM WITH NO ERRORS

```
1004Z0102311198000047380000200100491198O    3600000005004900000049000012R00
        360432100500Z                                        J1-     -      R
        320432100000Z                                        J1-     -      R
        320432400000Z                                        J1-     -      R
        320433100000Z                                        J1-     -      R
        320433600000Z                                        J1-     -      R
        230432304330Z                                        J1-     -      R
        140009900000Z                                        J1-     -      R
        460065601200Z                                        J1-     -      R
        460068001100Z                                        J1-     -      R
        220009904340Z                                        J1-     -      R
        260435000099Z                                        J1-     -      R
        380432100400Z                                        J1-     -      R
        491200000000Z                                        J1-     -      R
        230433504340Z                                        J1-     -      R
        490062000000Z                                        J1-     -      R
        210009904335Z                                        J1-     -      R
        490062000000Z                                        J1-     -      R
 MARCAT - PLACE THIS CARD AFTER YOUR LAST PROGRAM CARDFOLLOW WITH DATA R
0000123000060111 1
005555N0532112369
0000000066654321
       MARCAT II D OUTPUT
0000123000060111100000037500000000O
005555N05321J2369000556786R000000000 0
000000000666N432100361777860000000000
0 04  13 OK  23 OK  33 OK
```

TYPICAL STUDENT PROGRAM WITH ERRORS AND TRACE 1 PASS PROBLEM

```
1602Z0102311198000047380000200100491198O    3600000005004900000049000012R00
1602    360170800500Z                                        J1-     -      R
1602    130171700002Z                                        J1-     -      R
1601    2601000017l3Z                                        J1-     -      R
1602    120100000005Z                                        J1-     -      R
1602    2101000000992                                        J1-     -      R
1602    260170701000Z                                        J1-     -      R
1602    380170100400Z                                        J1-     -      R
1602    491200000000Z                                        J1-     -      R
 MARCAT - PLACE THIS CARD AFTER YOUR LAST PROGRAM CARDFOLLOW WITH DATA R
 250124
       MARCAT II D OUTPUT
 2680000250124000000000O

5 02  11 NO
0050036017080050O
00512130171700002          0124            02            000248
00524260100001713        000268        000025            000025
00536120100000005        000025            05            000020
00548210100000099        000020        000248            000268
00560260170701000        000268        000268            000268
00572380170100400
0058449120000000O

5 02  11 NO
```

THE 141 DATA PROCESSING SYSTEM
AN EDUCATIONAL COMPUTER FOR INSTRUCTION
IN ELEMENTS OF COMPUTER PROGRAMMING

Wilson T. Price
Kenneth P. Swallow

A basic course in the use of computers can serve many purposes. Undoubtedly the most important is to provide the beginner a firm, even if limited, foundation in stored programming concepts. The degree to which a beginning programmer grasps these basic principles will, to a large extent, determine his success in future programming classes. The choice of the computer, the programming language and the exercises to be used in the basic course are important factors in the ease with which stored program concepts can be imparted to the student. The 141 Data Processing System was designed solely as a vehicle for teaching these initial concepts. This system is an abbreviated version of the IBM 1401 Data Processing System and is an internally stored program machine with the following features:

1.	Input:	IBM Card Reader
2.	Output:	IBM Card Punch and 100 character per line Printer
3.	Storage:	1000 positions of core storage with three digit numerical addresses
4.	Instruction and data length:	Variable

Each position is designated by a three digit address in the range of 000 through 999 and is capable of storing one character: a letter of the alphabet, a numeric digit or a special character such as , / + or (. Internal character coding is very similar to the Hollerith code used in the punched card. A group of consecutive storage positions make up a field. A special indicator called a word mark is placed in the high order, or left most, position to indicate the length of the field. Both data and instruction fields are variable in length so that no storage space need be wasted with meaningless blanks or zeros. The instruction set of the 141 system consists of

1.	Move Characters to A or B Wordmark	MCW
2.	Set Wordmark	SW
3.	Clear Wordmark	CW
4.	Read a Card	R
5.	Punch a Card	P
6.	Write a Line	W
7.	Branch (conditional and unconditional)	B
8.	Compare	C
9.	Add	A
10.	Subtract	S
11.	Halt	H
12.	No Operation	NOP
13.	Clear Storage	CS
14.	Load Characters to A Wordmark	LCA

This set of fourteen instructions is sufficient to allow the coding of a wide range of programming problems, but at the same time it is small enough that the primary effort of the student is directed toward the understanding of programming concepts and not the memorization of a large number of operational rules. Simple exercises in 141 programming can illustrate such concepts as 1) looping for iterative processes, 2) sequence checking, 3) counted loops, 4) address modification, 5) program switches, 6) subroutine linkages, etc. Multiplication and division are accomplished through the use of 141 subroutines.

If coding a program builds confidence in the new programmer, seeing his program run and seeing it printed as it is represented in core storage can only strengthen that confidence and cement further the whole concept of stored programming in his mind. Any 141 program can be run on an IBM 1401, 1410 or 1460 Data Processing System. It can also be run on an IBM 1620 Data Processing System through the use of a special simulator program.

In addition to the concept of stored programming, a basic computer course should also include the introduction to a symbolic assembly language. The 141 system utilizes the IBM 1401 SPS (Symbolic Programming System) and 141 programs can be assembled on a 1401 or a 1620. The SPS pseudo-instructions are 1) Define Constant with Wordmark 2) Define Constant 3) Define Symbol and 4) Define Symbolic Address for area definition and 1) End 2) Origin and 3) Execute for processor control.

Because a large number of schools have IBM 1620 computers, program packages have been prepared for assembling and running 141 programs on the 1620. The programs and an accompanying manual have been submitted to the IBM Program Library by Kenneth P. Swallow and Richard Gentry, originators of the concept and are available in four versions to permit maximum utilization of hardware. The versions and their library file numbers are:

Non-Monitor Versions	File Number
Version A - Basic 1620	13.0.015
Version B - 1620 with 1443 Printer	13.0.016
Monitor Versions	
Version C - 1620 with 1311 and indirect addressing	13.0.017
Version D - 1620 with 1311, 1443 and indirect addressing	13.0.018

The 141 SPS Assembler makes it possible to process 141 programs on a 1620 computer. The resulting object deck is a listing deck and includes the original SPS program as well as loading instructions and the machine language program. This deck can then be loaded and run on either a 1401 or a 1620 with the simulator.

Using the 141 Simulator, it is possible to run programs on the 1620. The simulator has been designed so that the 1620 "acts" like a 141 system. Students have completed a full semester programming course without the need for studying the 1620 language or characteristics. Features of the simulator include a storage dump in the 141 machine language with 141 addresses, provisions for altering programs in the 141 language through the 1620 typewriter, provisions for stepping through a program a single 141 instruction at a time and a means for console debugging in the 141 language.

The 141 has been used for teaching high school students and college students in such varied disciplines as business data processing, accounting and engineering (pre-FORTRAN). It has, in every sense of the word, served our need for teaching fundamental programming concepts.