May 1, 1963.

NEWSLETTER

Committee on Computers in Research

For future issues of the Newsletter, we invite programmers to send in contributions of routines, programming techniques, or anything else of interest in this field. We also wish to receive, from economists and others, contributions relating to computer applications for economic and statistical research. In addition, we look forward to receiving news notes from members of the Committee.

To simplify distribution arrangements for this issue, we are sending 12 copies to each Bank. More copies are available on request.

Emil Melichar

CONTENTS

NEWS NOTES

Philadelphia

We have completed the programming of the X-9 version of the Shiskin seasonal method for our minimum configuration 1401. We are now in the process of comparing results with the Board's 1410 program. When thorough testing is completed, we will make the program available to any Reserve Bank that wants it.

The Bank is in the process of installing an 8K 1401 with multiply-divide and advanced programming features.

K. M. Snader

Dallas

Two general programs contemplated by the Federal Reserve Bank of Dallas include a general moving average program and a general least squares trend line. No estimate of the time of completion of these projects has been made.

John D. Stuligross

Atlanta

Data Processing Department (Jeff Wells) is presently writing a program for editing and processing data from FR 209--Report of Examination, Statistical Data. When completed, this program will be made available to other Federal Reserve Banks.

We are making a study of charting procedures that might be feasible for our machine. We are especially interested in a program that will plot a tier chart by months for several years.

We are currently programming extensive tabulations of call report data for all insured banks. These will be used in several articles dealing with agriculture and with bank structure.

We are making increasing use of the printer to produce final copy for releases, etc., for which plain, white paper is needed in reproducing. We have found that the back of standard one-part paper of any type gives very good results.

We have received a three-tape sort program for the 1401. Configuration required is 4-K 1401 machine, advanced programming, high-low-equal compare, sense switches, and three tape units. The program's function is to sort a file of tape records into ascending order using three tape units. It was written by R. Zuidhof of a Dutch oil firm and was obtained through the IBM program library.

W. M. Davis

Boston

Here in our Boston Research Department we do programming for both our own IBM 1620 which is a 20K machine and a 4K IBM 1401 located in our Central Tabulating Department. The 1401 has recently been purchased.

Current and recent programming efforts include the following:

1. Income and Expense reports for some 200 member banks are to be balanced, edited, analyzed and compared with averages from the four Call Reports for the year.

2. Our Trust Survey analyzes Trust Department Income and Expense. The program will compute individual ratios for each back and average ratios for banks grouped by size of Trust Department and size of various deposit categories.

3. The Functional Cost Analysis which was recently completed gives each of the 91 participating banks a rather complete breakdown of the banks profitability by function on comparisons with an average of similar banks. This program resulted from the combined efforts of our own Research programming personnel, programmers from Central Tabulating and Planning Departments, plus C.E.I.R., an outside consulting firm. Substantial parts of this program will be used by the New York bank for a similar report which they do for their member banks.

4. The New England Camping Report, programmed recently, gives percentage changes from last year for July and August camp registrations. This is done according to five major breakdowns and various categories within these.

Howard G. Smith

St.Louis

The Planning Department is writing an all-purpose program to take time series and compute indexes, first differences, etc. (lots of manipulations) and prepare output in a form compatible with statistical program formats (i.e., regression, seasonal adjustment).

Willard T. Carleton

Cleveland

We have recently completed a revision and rewriting of the Kansas City F.R.B.'s regression program for an 8K 1401 tape system equipped with 4 tape drives and advanced programming features. The revised version, while not altering the program logic or the mathematical methods, eliminates the intermediate card output and supplies labels identifying all output items.

We have finished writing three programs in connection with a special survey of short-term investments of state and local governments in our District.

We are planning to program the Census II method of monthly seasonal adjustment, including the X-9 and X-10 modifications, for 8K 1401 equipment, using tape and advanced programming. Currently we are at the point of developing the mathematical formulas for the computations. Would anybody who has the sequence of mathematical steps volunteer to let us "borrow" them?

We have found out from IBM--as other Banks undoubtedly have, too--that a revised Autocoder (2/19/63) is available.

Gerhard Krebs

Kansas City

We have several current programming projects at our Bank:

1. We are working on programs to compile, on a per bank basis, Federal funds activities and discount positions. These are reports for internal management use that have been compiled by hand in the past.

2. A regression program is being worked on that will accommodate numerous input data formats and that will perform calculations in floating point.

3. A program is being prepared that will print and also update routing slips for our research library.

4. Our Condition and Income and Dividends edit programs have been changed so that the computer, instead of the key punch operators, round the data. Also, the computer now handles the duplication of the cards to be sent to the Board.

Slip sheets containing the Operating Ratio headings and individual bank ratios are now entirely printed by our computer.

Edwin F. Terry

FORTRAN NOTES

A problem was noted in the ForTran compiler, Version 2 level 1, at the Federal Reserve Bank of Dallas. An improper subscript of an array was not noted in the error listing and the program compiled, but, of course, did not run properly. The instruction given was:

If (Icode - 9) 9, 10, 9

Icode was an array.

John D. Stuligross
Federal Reserve Bank of Dallas

I had a ForTran program I was trying to compile on their Version 1, Mod 6, 1401 compiler. Computed subscripts, bounded by "DO" loops were exceeding the storage capacity of our computer and preventing my use of the program. Upon receiving the Version 2, Mod 0 compiler, the identical ForTran source program compiled with no difficulty at all. The Version 2, Mod 0 was apparently available the last of 1962. I received a copy from an IBM customer in Kansas City in February, through the efforts of a local IBM man.

Edwin F. Terry
Federal Reserve Bank of Kansas City

One of the economists in Research has just finished writing a ForTran program for an econometric model of the U.S. (from a pattern developed at the University of Michigan) but is discovering that a ForTran program involving 32 simultaneous equations refuses to assemble on an 8K machine.

Gerhard Krebs
Federal Reserve Bank of Cleveland

Editor's Note: The above program was assembled and run on the Board's 1410 computer in the 16K, 1401 mode.

MACRO-INSTRUCTIONS FOR THE 1410 AT THE BOARD

The purpose of developing a set of macro-instructions at the Federal Reserve Board was to make possible Autocoder programming by a wide range of people at the Board with varying backgrounds and capabilities. This was a necessary step in maintaining open shop programming since the simplified systems of programming developed by IBM (ForTran and Cobol) proved to be very slow on the 1410, both in compilation and at object time when the program was being executed. The reasons for developing macro-instructions were thus to simplify programming for part-time programmers, to minimize careless programming errors by making as much as possible automatic, and finally to provide a powerful set of summary instructions for the most complicated type of mathematical programming for the more experienced and sophisticated programmers at the Board.

Each of the 32 macro-instructions can be written in one line on the programming sheet, and all the necessary 1410 instructions to accomplish the macro job will be introduced automatically in one's program when the standard Autocoder assembly run is made. A series of programmed halts within the macros helps to protect against programming or data errors.

Fixed Point Arithmetic and Tracing

DIVFR* - does division of variable length fixed point numbers, scales and rounds to the number of decimal places specified and places the quotient in the specified answer area.

*Editor's note: The following example of the division macro may be informative: If I want to divide FIELDA (a whole number) by FIELDB (a whole number) and get the answer to 2 decimal places in FIELDC, I write one instruction:

```
DIVFR    ÷ 2,FIELDB, FIELDA,FIELDC
```

and I know that the result will be in FIELDC, properly rounded off and with the right sign. No setting up of working areas, etc., is needed. The rest of the macroswork in the same general fashion.

DVTFR - does division as described above and also prints a line for program testing purposes which show the addresses and contents of the locations for the dividend, divisor, and quotient.

MPYFR - does multiplication of variable length fixed point numbers, scales and rounds the answer the number of places specified, and places the product in the specified answer area.

MPTFR - does multiplication as described above and also prints a line for program testing purposes which shows the addresses and contents of the locations for the multiplicand, multiplier, and product.

ADDFR - does addition of variable length fixed point numbers and also prints a line for program testing purposes which shows the addresses and contents for the two locations being added and the location where the sum is to be placed.

LOGFR - computes the logarithm of a number to a requested number of decimal places and calculates the characteristic according to the decimal places specified in the input.

ALGFR - computes the antilogarithm of a number according to the number of decimal places in the input and the number of decimal places specified for the answer.

MAVFR - computes any specified term moving average of an array of data.

General Utility Routines

RPTFR - makes automatic the address modification, counting, and controlling within a loop in one's program. (Up to 3 index registers)

BLKFR - blanks out a specified section of core memory.

CP1FR - moves large blocks of memory within core including wordmarks.

CP2FR - moves large blocks of memory within core excluding wordmarks.

HKPFR - checks the status of the arithmetic and division overflow indicators and prints a message indicating the status of these indicators. Used at the end of the program, it provides assurance that overflow did not occur during production runs.

Floating Point

FLTFR - converts a number from fixed point to floating point form according to the decimal specified in the instruction.

FIXFR - converts a number from floating point to fixed point form and scales and rounds according to the number of decimal places specified in the instruction; will edit the number with decimal point and commas if specified in the instruction.

FPAFR[1/] -floating point addition of two numbers.

FPSFR[1/] -floating point subtraction of two numbers.

FPMFR[1/] -floating point multiplication of two numbers.

FPDFR[1/] -floating point division of two numbers.

SQRFR - takes the square root of a floating point number.

Input-Output

RCDFR - reads a punched card, with transfer to a final sequence of instructions when the end of file card is read.

PCHFR - punches a card.

WRIFR - prints a 132 character line.

1/ All of the floating point arithmetic routines are three address instructions which carry ten digits of precision in every answer and round after every operation. Like the fixed point arithmetic macros, they contain an option which permits tracing so that each time the instruction is executed, a line is printed with the addresses and contents of the three operands in the instruction.

2/ Unlike the 1401 computer, each input-output instruction on the 1410 requires the checking of 6 status indicators which indicate various possibilities of machine status such as input-output units busy, machine error was made, or data input is invalid, etc. This is all accomplished automatically by the macro input-output instructions. Tape and disk instructions require similar checks on the 1401.

CCIFR - Carriage Channel Control Instruction.

WCPFR - writes a message on the Console Typewriter.

WTPFR - writes a tape block.

RTPFR - reads a tape block.

TPUFR - checks status indicators after a tape unit instruction.

Special Input-Output

WR2FR - automatically edits and prints out a line of 11 ten-digit field.

WAEFR - automatically edits and prints out a number of fields once the mask (set of edit words) is specified by the programmer.

RTOFR[3/] - reads magnetic tape in the overlapped mode.

ROSFR[3/] - check status indicators after an overlapped tape read.

Contributed by Robert M. Steinberg,
Division of Data Processing,
Board of Governors.

Editor's note: I can personally testify to the great value of the above macro-instructions in making 1410 programming simpler and more accurate. Similar instructions can be written for the 1401 Autocoder. It seems to me to be very pertinent to (1) request a report for the next newsletter on any such instructions in use in the 1401 Autocoder and (2) suggest that the Committee at its next meeting consider sponsoring an effort to program some standard 1401 macro-instructions.

At the SPS level, a similar approach is the writing of standard subroutines that are incorporated in source decks. The Philadelphia multiplication and division routines, the Richmond division worksheet, and the Richmond Romac input-output routines are examples in use in the System.

3/ The use of these two macros permits the programmer to automatically overlap the computation on a tape record with the reading of the next tape record so that the computer will not be interlocked while the tape reads in.

INDEXING TECHNIQUES

Indexing saves a lot of programming time and also the core storage that would have been used by instructions to modify addresses without indexing, but core storage is still at a premium in many 1401 programs even when indexing is used. Here are some techniques that will save core space in the process of manipulating index registers:

1) An index register can be cleared to zeros by subtraction if you address the next location to the right of the register. With a word mark in the high-order position of each register, the following 4-digit subtract instruction will clear the register to zeros without leaving AB-bits in the units position:

Index register	Clear by this instruction	
	SPS	Autocoder
1	S 0090	S 90
2	S 0095	S 95
3	S 0100	S 100

2) It is possible to add to an index register without setting a word mark in the high-order position (as would be required on 4K machines--the MA instruction used on larger machines does not need a word mark) and without setting up a constant containing the number to be added to the register. This opens the door for significant savings in core storage on all sizes of machines. It is accomplished by using a seven-digit SBR instruction as follows:

Op code: SBR
A-operand: low-order address of index register being added to.
B-operand: the actual number to be added to the register, indexed by the register being added to.

Example: Add 19 to index register 1.
SPS Instruction: SBR 0089 0019 1
Autocoder Instruction: SBR 89,19+X1
Assembled Instruction: H0890/9

An index register can also be cleared to zeros by the SBR instruction.

Example: Clear index register 1 to zeros
SPS Instruction: SBR 0089 0000
Autocoder Instruction: SBR 89,0
Assembled Instruction: H089000

An actual number can also be put into an index register by the SBR instruction.

Example: Put the actual number 2,723 into index register 1.
SPS Instruction: SBR 0089 2723
Autocoder Instruction: SBR 89,2723
Assembled Instruction: H089P23

The first saving is achieved in not having to set up a constant containing the number to be added to the index register. This both saves core storage and avoids the common programming error of forgetting to set up the constant. Second, the elimination of the need for a word mark for the index registers makes it possible to use the techniques described below on 4K as well as on larger machines.

3) If a word mark is set only in the high-order position of index register 1, two or three index registers can be cleared to zeros with one 4-digit subtract instruction. The instructions used are as follows (remember to have a word mark only in location 087):

To clear these index registers	Give this instruction SPS	Autocoder
1	S 0090	S 90
1 and 2	S 0095	S 95
1,2,and 3	S 0100	S 100

The key to efficient programming using this technique is to assign the index registers in the "correct" order. In general, assign index register 1 to the use that will require it to be cleared most often, and index register 3 to the use requiring clearing least often (i.e., at the end of the larger loops). Look for opportunities to clear register 1 and 2 or registers 1, 2, and 3 at the same time. When this technique is being used, the SBR instruction cited in (2) above can be used to clear index register 2 or index register 3 without affecting the other registers. In addition index registers 2 and 3 can be efficiently cleared together by the following chained SBR instructions, which will place zeros in locations 091 to 099:

SPS	Autocoder
SBR 0099 0000	SBR 99,0
SBR	SBR
SBR	SBR

4) In many programs, the programmer could benefit from having more than three index registers. Suppose all three are in use when you come to an instruction loop in which you could use three more, but temporarily do not need what is already in the registers. With a word mark in 007 only, it is possible to store the present contents of all of the registers with one instruction, clear all the registers with a second instruction, write the loop, and then restore the old contents of the registers with just one more instruction. The sequence is as follows:

SPS	Autocoder
MCW 0099 STORE1	MCW 99,STORE1#13
S 0100	S 100
.	.
.	.
.	.
Loop	Loop
.	.
.	.
.	.
MCW STORE1 0099	MCW STORE1,99
:	
.	
.	
13 STORE1 DCW	

5) If the above techniques are to be used several times in a program, they should be incorporated in subroutines to save more core storage. Suppose, for example, you have a program in which you want to use the equivalent of six index registers, three at a time, and you want to switch back and forth from one set of three to the other without destroying the contents in the process. This can be handled by setting up two storage areas in which to save the contents of the index registers, and two subroutines to move the contents back and forth. The sequence of instructions and the subroutines might be as follows:

SPS	Autocoder
.	.
.	.
.	.
Program	Program
.	.
.	.
.	.
B SWICH1	B SWICH1
.	.
.	.
.	.
More Program	More Program
.	.
.	.
.	.
B SWICH2	B SWICH2
.	.
.	.
.	.
More program	More program
.	.
.	.
.	.
B SWICH1	B SWICH1
.	.
.	.
.	.
More program	More program

```
SWICH1  SBR  RET1 +003          SWICH1  SBR  RET1+3
        MCW  0099 STORE1                MCW  99,STORE1#13
        MCW  STORE2  0099               MCW  STORE2#13,99
RET1    B    0000               RET1    B    0

SWICH2  SBR  RET2 +003          SWICH2  SBR  RET2+3
        MCW  0099 STORE2                MCW  99,STORE2
        MCW  STORE1  0099               MCW  STORE1,99
RET2    B    0000               RET2    B    0
```

```
13  STORE1  DCW
13  STORE2  DCW
```

Contributed by Emil Melichar,
Division of Research and Statistics,
Board of Governors.