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: H0390/9

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

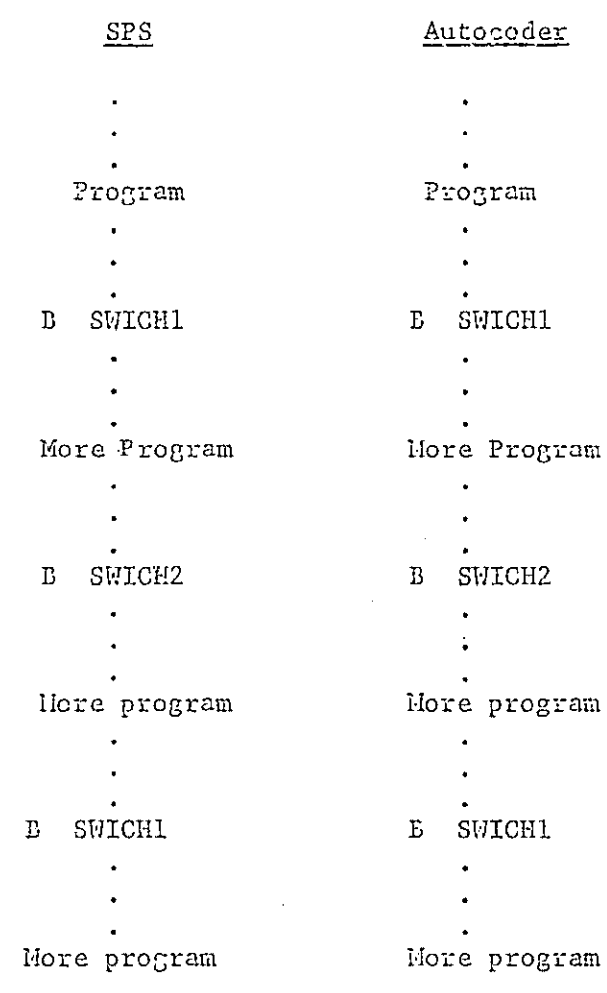
The key to efficient programming using this technique is to assign the index registers in the "correct" order. In general, assign index register 1 to the use that will require it to be cleared most often, and index register 3 to the use requiring clearing least often (i.e., at the end of the larger loops). 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 037 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:




```
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.