Self-Modifying Code for the TI-58/59

Ted Green, Box 2289-AMR
Johns Hopkins University
Charles and 34th St
Baltimore MD 21218

Because of the four multiregister memories in the Texas Instruments TI-59 programmable calculator and their ability to hold either data or program steps, it is possible to let the TI-59 change its set of instructions, or any segment of its instructions, at any time during the program. This is done by "overlapping" data registers and program steps.

To see how the TI-59 stores numbers contained in the data register in the program-step memory, enter the following, repartitioning to 100 data memories, 0 steps:

```
1234567891
STO 99
0
Op 17
GTO 000
LRN
```

Examine the *LRN* mode using *SST*; keep in mind that originally there was nothing in the *LRN* mode. Now, we examine the following locations:

```
000 90
001 00
002 00
003 91
004 78
005 56
006 34
007 12
```

The code in location 000 represents the type of number that was entered. In this case, the 9 stands for a number that consumed 9 memory locations (location 007 represents memory location 1, location 6 represents memory locations 2 and 3, location 5 is for memory locations 4 and 5, etc). Notice that the number entered as 1234567891 is stored as 9178563412 (starting at location 003). The empty registers 001 and 002 are used for the storage of up to thirteen digits (in location 001, the rightmost digit is always 0). If you entered 1234567891 and stored it in data register 98, your *LRN* mode would look like this:

```
000 00   008 90
001 00   009 00
002 00   010 00
003 00   011 91
004 00   012 78
005 00   013 56
006 00   014 34
007 00   015 12
```

Storing the same number in data register 97 would use memory locations 016 thru 023, and so on. This scheme continues throughout, with data register 00 taking up memory locations 952 thru 959.

To apply this principle, try the following example:

9
Op 17
8166950185
+
.686
=
STO 99
0
Op 17
RST

Now examine the *LRN* mode and notice the following:

000 90 List
001 60 Deg
002 68 Nop
003 85 +
004 01 1
005 95 =
006 66 Pause
007 81 RST

This is a counting program. Press *RST*, *R/S*, 1 . . . 2 . . . 3 . . . 4 . . . etc. The .686 was added because neither the *Deg* nor the *Nop* have any effect on numbers that are "carried" from one step to another.

There are drawbacks to this storage system. For instance, if the number 1 is stored in memory 99, all program locations 001 thru 006 are cleared, erasing everything between 000 and 007. Also, the instruction 000 90 appears to be troublesome and cannot be changed to a useful code; all it does is take up space. In addition, the code in 002 always has a 0 on the rightmost side, which disables the code. Keep in mind that this also applies to codes 008 and 009, 017 and 018, all the way up through 952 and 953.

Listing 1 is an actual program that will first begin as a counting program, then, after adding 1, it will modify its instructions so that it becomes a subtraction program. ■

Listing 1: *A demonstration program showing self-modifying code on the Texas Instruments TI-58 or TI-59 programmable calculators. When run, the program adds 1 to the number on the display, then continually subtracts until* R/S *is pressed. Begin execution at step 950. As soon as the program begins, hold down the Pause key to see the program work. After the program has been run, examine the* LRN *mode to observe how the code has been modified.*

Step	Code	Key
000	76	Lbl
001	12	B
002	05	5
003	69	Op
004	17	17
005	01	1
006	01	1
007	06	6
008	01	1
009	09	9
010	05	5
011	00	0
012	01	1
013	07	7
014	05	5
015	85	+
016	93	.
017	06	6
018	08	8
019	06	6
020	95	=
021	42	STO
022	00	00
023	00	00
024	69	Op
025	17	17
026	61	GTO
027	09	949
028	49	—
.	.	.
.	.	.
949	32	x≷t
950	76	Lbl
951	11	A
952	85	+
953	01	1
954	95	=
955	32	x≷t
956	61	GTO
957	12	B