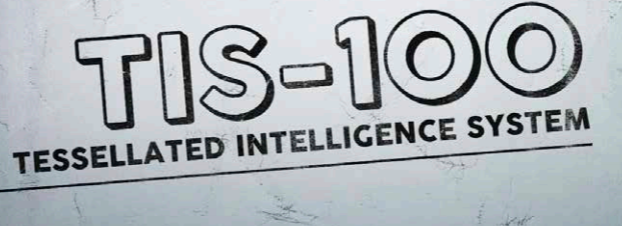
(Approx. 728 words)

## TIS-100

By Dick Maybach, Member, Brookdale Computer Users Group, NJ

May 2019 issue, The Monthly Newsletter of the Brookdale Computer Users Groupwww.bcug.com

n2nd (at) att.net



At the start of the PC era, the devices were much simpler, and a major attraction for their enthusiastic owners was writing software. TIS-100 is a game that lets you return to those earlier, simpler times. It’s most readily available for Linux, Macs, and Windows for $7 through Steam, <https://store.steampowered.com/app/370360/TIS100/>. (You will also need the steam client, but this is free.) It presents you with a series of programming problems for a three-by-four array of very simple processors. There are just 14 documented assembly-language instructions (although the player may find additional undocumented ones as play progresses), and only storage is provided by two registers in each processor. In later problems, the hardware is augmented with a simple display and stack storage. One or more of the processors have failed, and this changes for each problem.

The first time you start, you will see the display of Figure 1; follow the advice and print the manual.



Figure 1. Introductory TS-100 Screen.

The first page of the manual summarizes the background story, as shown in Figure 2.

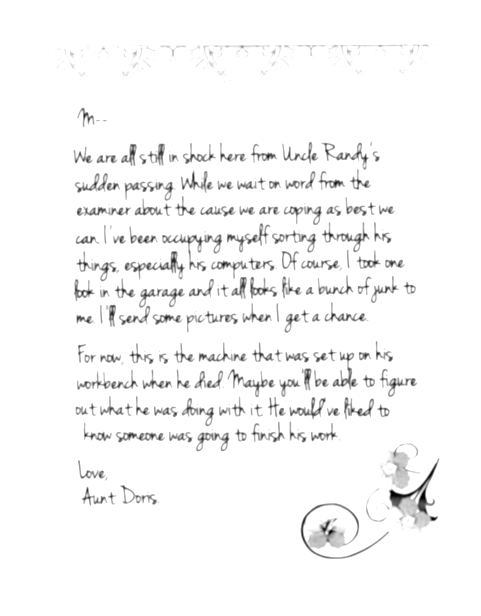


Figure 2. Game Background.

The bulk of the 14-pages contain a description of the instruction set for the TIS-100 computer mentioned in the note and is essential, since the game is a series of problems to program the computer. However, there are no instructions on how to play the game, and it can be frustrating to get started.

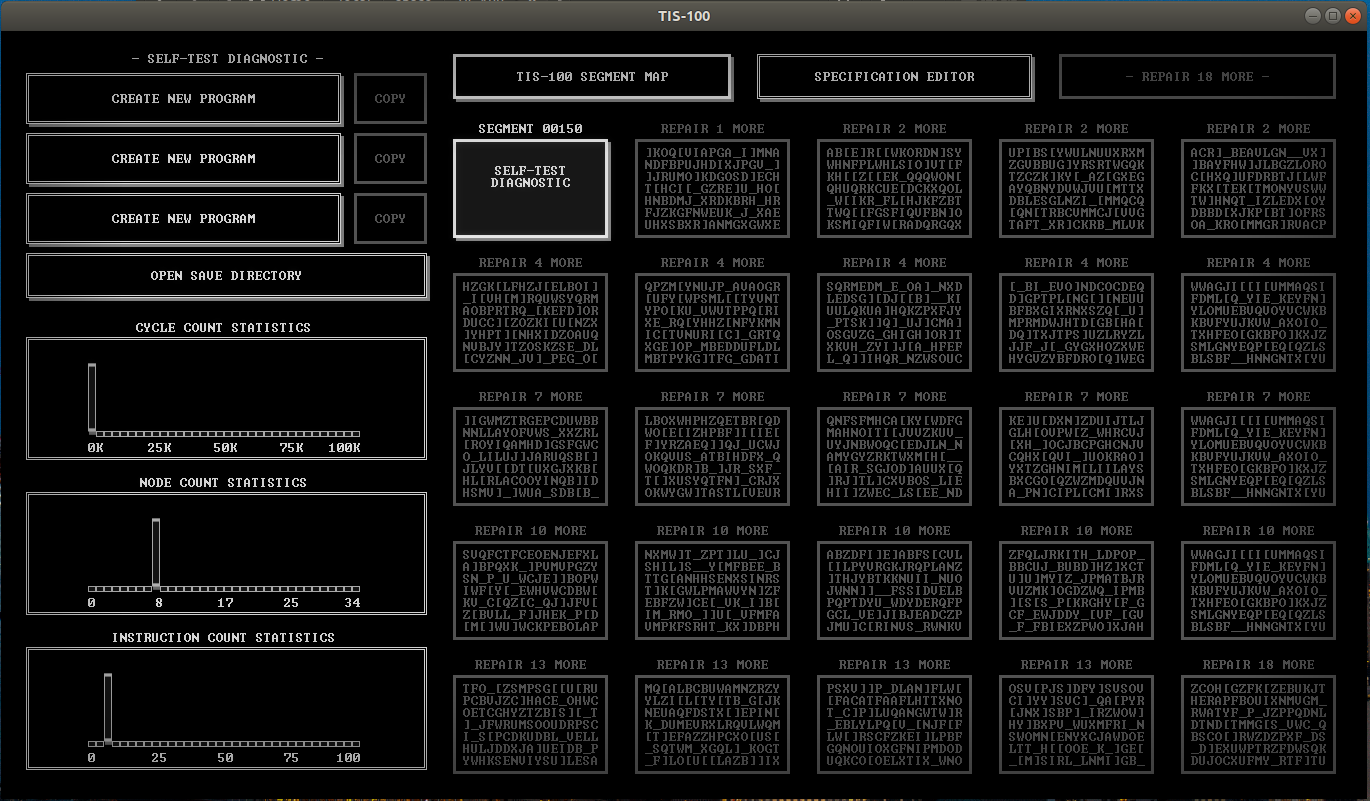


Figure 3. TIS-100 Opening Screen.

Except for the first time you run the program, Figure 3 is the start-up screen. Each of the rectangles in the right portion is a different problem, and initially you have only one choice, a self-test diagnostic, although the number of choices increases as you proceed.

You begin by selecting the box labeled “Create New Program” at upper left, and you will get your first look at the TIS-100 computer, Figure 4.

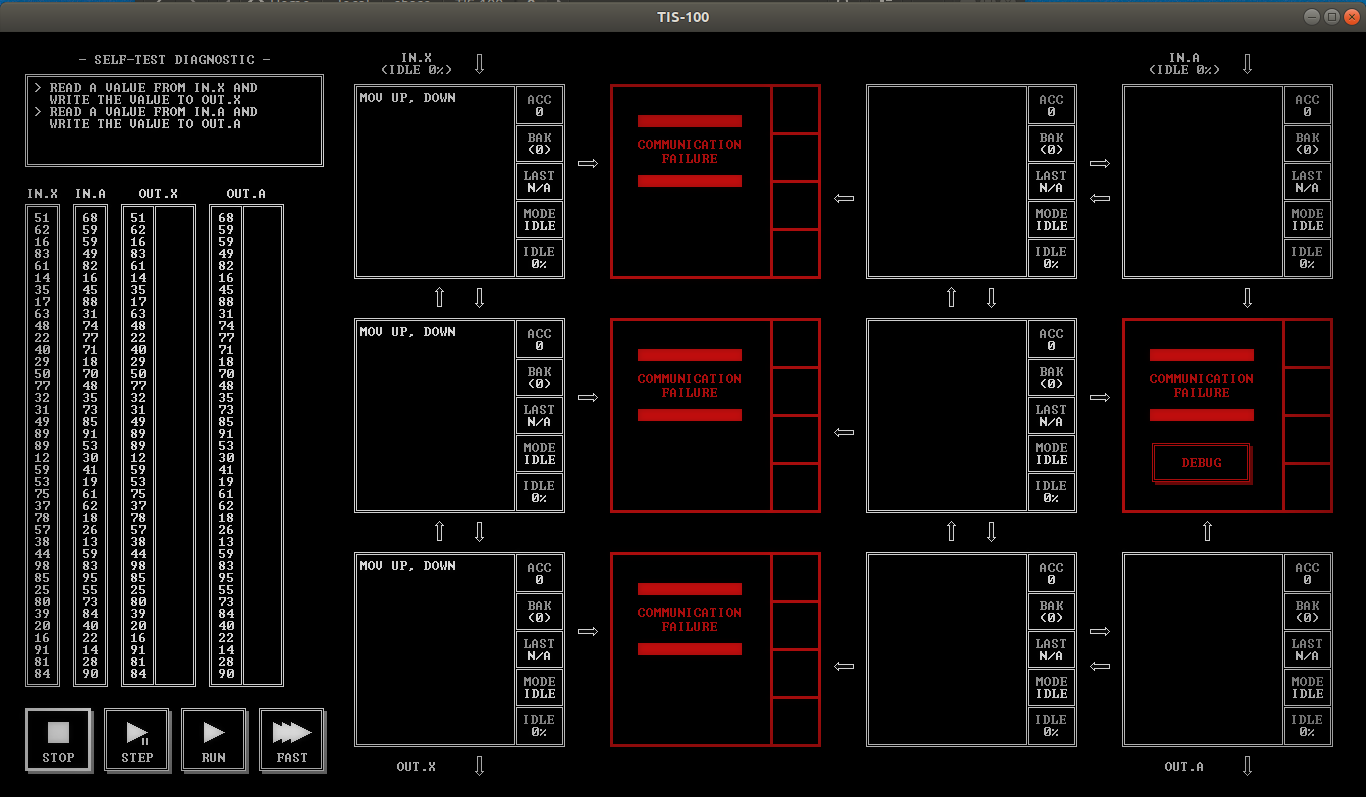


Figure 4. TIS-100 Self-Test Diagnostic Problem.

As you can see, the computer is a three-by-four processor array; however, several of them (shown in red) have failed, and you must work around them. The particular units that have failed differ for each problem. Note the box marked “Debug” on one of the failed units. Selecting it will show you an entry from a notebook of the deceased Uncle Randy, who was working on the TIS-100 before he died. Almost every problem has one of these, but so far, I haven’t found anything that helps in finding solutions.

The upper-left box shows a statement of the problem, and this is expanded in Figure 5.

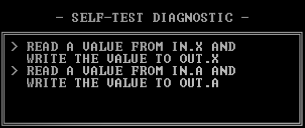


Figure 5. Self-Test Diagnostic Problem.

IN.X is an input to the top left module, and OUT.X is an output from the bottom left one. All three processors in the left column have already been programmed to perform the task of moving a number from IN.X to OUT.X. Your problem is to program the machine to do the second task, which is complicated by the failed module in the column on the right.

Below the problem box is data to test your solution, which in this case is input to IN.X and IN.A and checked against the OUT.X and OUT.A from your solution. You can single-step through this if needed. Each processor displays its state, as shown in Figure 6.

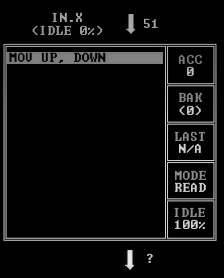


Figure 6. Processor State.

Figure 7 shows debugging in process.



Figure 7. Self-test Diagnostic running.

After a problem ends, you see a display of the resources it used, which you can use to rate your different solutions. (The game allows you to create more than one solution to a problem.)

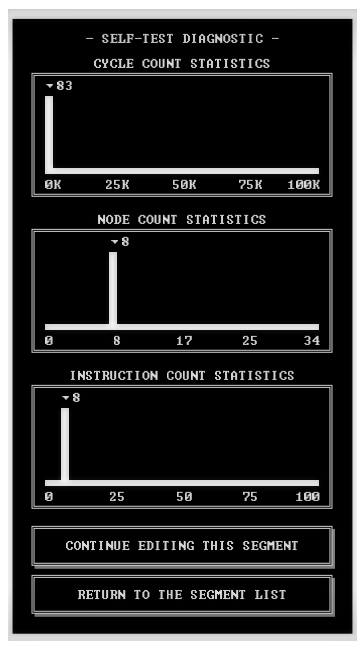


Figure 8. Solution Statistics.

The example I’ve shown here is of course trivial, but the complexity quickly increases as you move on to harder problems, especially if you try to optimize your results. For example, if data is moving through more than one path, keeping things synchronized can be challenging. Arrays of very simple processors can accomplish surprisingly complex tasks.

If you enjoy this, a similar game Shenzhen I/O by the same designer is also available from Steam for Linux, Macs, and Windows, <https://store.steampowered.com/app/504210/SHENZHEN_IO/>. It includes hardware as well as software design. Working with these games will make you appreciate how much early researchers accomplished with the simple hardware they had.