

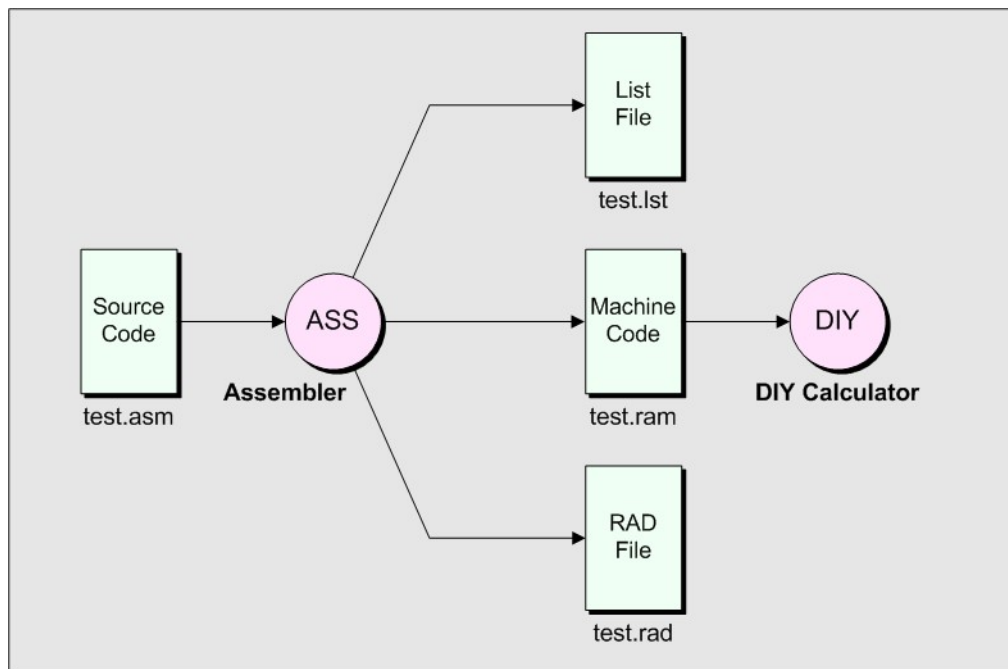
# DIY Calculator: Data Gatherer

## Introduction

The purpose of this paper is to document a new “Data Gatherer” utility that has been added to the DIY Calculator’s CPU.

## An Overview of the Flow

When we run our assembler, it uses a *.asm* (source code) file as input and generates three files as output: a *.lst* (list) file, a *.ram* (machine code) file, and a *.rad* (raw assembly dump) file. For example, consider what happens when we assemble a program called *test.asm*:

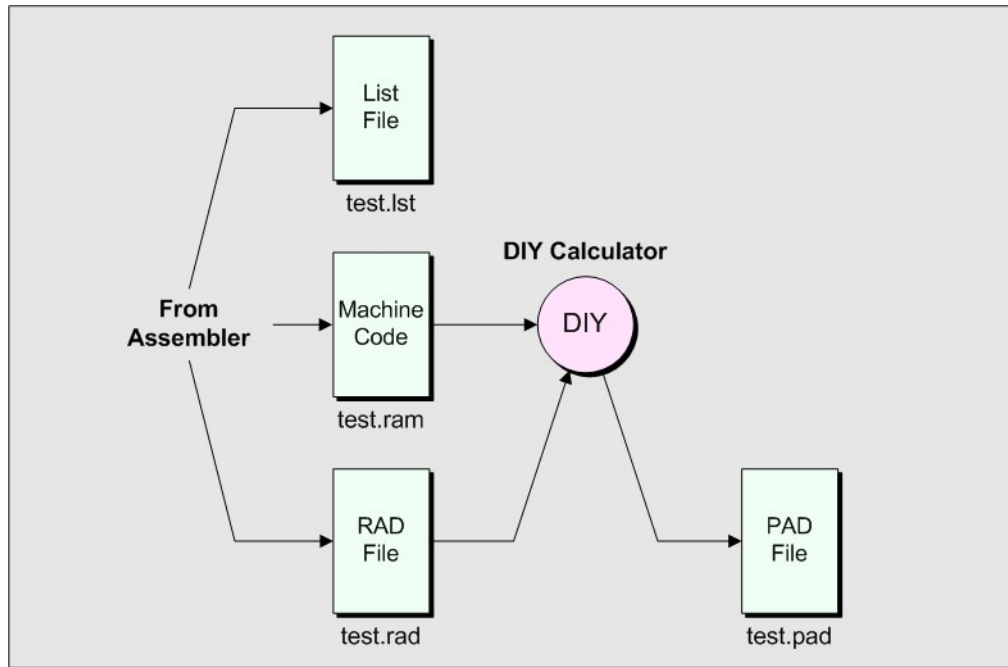


As we know, we can load a *.ram* (machine code) file into the DIY Calculator’s memory by means of the **Memory > Load RAM** command, at which point we can run this program. Thus far, we’ve used the assembler’s list file output only to debug our programs by stepping through the program and visually examining these files, and we haven’t used the Raw Assembly Dump (*.rad*) files at all, but that’s about to change.

The point is that we have augmented the DIY Calculator with a “Data Gatherer” utility, which can be used to gather data during the course of a program run. This version of the calculator can be downloaded from the **Download** page of the DIY Calculator website at [www.DIYCalculator.com](http://www.DIYCalculator.com).

The type of data we’re talking about here is the number of times each opcode is executed and the number of times each data location (reserved using a `.BYTE` command or its `.2BYTE` and `.4BYTE` cousins) is written to and/or read from. In the case of conditional jump instructions such as a `JZ` (“jump if zero”), the Data Gatherer utility will record how many times the instruction passes its test and how many times it fails.

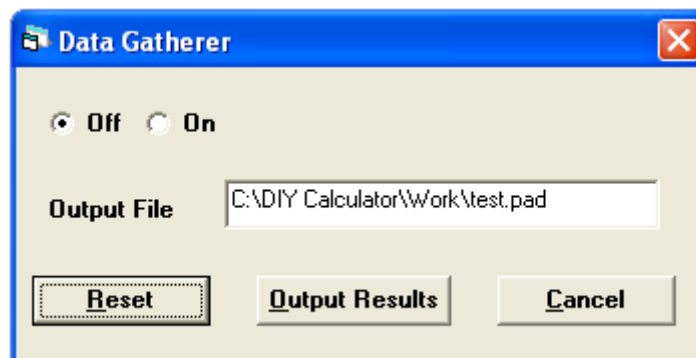
The way in which the Data Gatherer utility works is presented later in this document. For the purpose of this portion of our discussions, we need only be aware of the fact that – when we instruct it to do so – the Data Gatherer can be used to read in the \*.rad (raw assembly dump) file generated by the assembler and to write out a corresponding \*.pad (processed assembly dump) file that contains the gathered data:



Note that the formats for each of these files are fully documented under the **File Formats** topic on the **More Tools** page of the DIY Calculator website at [www.DIYCalculator.com](http://www.DIYCalculator.com).

## Using the Data Gatherer

Using the Data Gatherer is really easy (actually using the data it gathers presents more interesting problems as discussed below). First we use the **Tools > Data Gatherer** command to launch the following dialog:



When you use the **Memory > Load RAM** command to load a machine code file into the DIY Calculator's memory (say the *test.ram* file as discussed in the previous section) then the **Output**

**File** field will be automatically populated with a corresponding *test.pad* filename as shown in the illustration above. Note, however, that you may edit this filename as you wish; for example, you may wish to perform several data gathering runs, and save these out with names such as *test-v1.pad*, *test-v2.pad*, *test-v3.pad*, and so forth (irrespective of the output filename, the Data Gatherer will always continue to use the original \*.rad file as input).

When you click the **On** radio button on the Data Gatherer it will start collecting data; when you click the **Off** radio button it will stop; and when you click the **Reset** button all of the gathered data values will be reset to 0 (zero). A typical way of using this would be to load a machine code program and click the **On** button on the Data Gatherer; nothing will happen until you click the **Run** button on the DIY Calculator to start running the program. Once you do click the **Run** button the Data Gatherer will start gathering data. When you've finished gathering data, you would click the **Reset** or **Step** buttons on the DIY Calculator, click the **Off** button on the Data Gatherer, and then click the **Output Results** button to output your gathered data in the form of a PAD file.

Another common scenario would be to start with the Data Gatherer in its **Off** state. After loading a machine code program, you might then set one or more breakpoints in the Memory Walker display (which you invoke using the **Display > Memory Walker** command). Next, you may run to the first breakpoint, at which time the DIY Calculator will automatically drop back into its **Step** mode. Now you may click the Data Gatherer's **On** button and then click the DIY Calculator's **Run** button to run to the next breakpoint. When you reach this breakpoint, you may click the Data Gatherer's **OFF** button and then output your gathered data in the form of a PAD file. One use for this second scenario might be to gather the data associated only with a particular portion of your program such as a math subroutine.

## What Will We Do With the Gathered Data?

If you visit the **More Tools** page on the DIY Calculator website at [www.DIYCalculator.com](http://www.DIYCalculator.com), you will find discussions on creating Code Coverage and Code Profiler utilities.

The idea here is that somebody [maybe you, maybe us] will create these tools, which will read in the data gathered by the Data Gatherer as discussed above. They will then analyze this data and present it to us in such a way as to facilitate our investigating the ways in which our programs are running.