

# **New Release of Simulator Available with Script ECE 662**

**Prof. C. Klein**

**(Modified for Autumn Quarter 2007 by Prof. D. Orin)**

The last release of the ECE 662 simulator had two limitations that have now been overcome. The last release required one `memory.dat` file for each test program. The new release features a new format for this file that permits multiple tests in a single file. This new release also allows you to specify an initial value for the general purpose registers before the first instruction is executed. In this way, many tests which used to require several instructions to set up, can now be done in one instruction. For example, to test the jump to subroutine instruction, the stack must be initialized. By presetting the SP, it is no longer necessary to do this with instructions.

The second change is that a script file is available to run the program. This script accepts the logical names for the microinstruction and test program files and automatically makes copies into the names `specs.dat` and `memory.dat`, respectively. Therefore, users can make multiple versions of the input files without having to explicitly name them constant names.

## **Format of Test Files (old `memory.dat` files)**

The most extensive change the user sees in this new release is the format of the `memory.dat` file. In order to make the change relatively simple, the file still has a fairly rigid format.

All lines in a `memory.dat` file fit one of two formats. The first kind of line is one which has a valid hex number in the first 4 columns. Call this type of line a hex format line. The other kind of line is one that does not have a valid hex number in the first 4 columns. Call this second kind of line a delimiter line.

The first rule of `memory.dat` files is that they consist of 1 or more test sets. A test set is a contiguous set of hex format lines. The lines of a test set describe the original values of the AC, X, SP, PC and Condition Code registers, and the successive words of memory starting from address 0.

The values of the AC, X, SP, and PC registers and the memory contents are 4 digit hex numbers. The four condition codes are put in one 4 digit number interpreted as a 4 bit binary number. The bits of this field are assigned from left to right to the CVZN bits, respectively.

The second rule of this file format is that any number of delimiter lines can appear before, after, or between test sets, but not within a test set. At least one delimiter must appear between test sets as a separator. Additional ones can serve as comments.

Spaces have a predefined interpretation by the Linux operating system. Spaces can be used in place of leading zeros in the 4 digit hex numbers, but the number must be right adjusted in the first four columns. A blank line is treated as a hex zero and cannot be used as a separator between test sets.

Anything past the first four columns is ignored and therefore, can be used as comments. Therefore, it is possible to comment the text instructions within the lines of the test.

Here is a sample `memory.dat` file:

```
-----cut here-----
this is a sample memory.dat file
starting test 1
0001 AC
0002 X
   FF SP
0000 PC
1100 CVZN
1001
0000
this is end of test1
*****
this is test 2
FFFF AC
FFFE X
   FF SP
0000 PC
0000 CVZN
1001 ADD AC,X
0000
this is end of test2
-----cut here-----
```

This sample file has two tests. Suppose 1001 is the instruction to ADD AC to the X register. The value of 0 in location 1 is the halt instruction. In the first test, by presetting the AC and X registers to 1 and 2, one can look for the answer 3 in the

X register. Without the ability to preset registers, tests would have to be longer. For example, for this test, it would have required additional instructions to generate non-zero values to add. A second part of this first test is to see if the initial values of 1 for C and V are properly cleared, since no carry or overflow is generated in this add.

Test 2 is the same except for the operands. Here we should expect to see -3 in the X register (FFFD Hex) and the N condition code set.

The format of this file is, unfortunately, deceptively simple. In this example, the characters “AC” in the first line of the test set are actually a comment. Putting the AC value here is not an option since the first 5 lines are always the values to go into the AC, X, SP, PC, and condition codes, whether the comments are included or not. However, I strongly recommend the comments especially to remember the order in which the condition code bits must be presented.

All test sets must have the first five lines to preset registers and condition codes, even though in most cases all zeros is sufficient. It should be easy however, to make a skeleton test file with these 5 lines already included. By copying this skeleton to a different name and filling in the actual instructions in the test, you can save a lot of typing. By including the comments “AC, X, SP, PC, and CVZN” in this skeleton file, you will be able to read the resulting test file a great deal easier.

### **Microinstruction File (old specs.dat file)**

No changes have been made to the format of this file.

### **Output Files Changes**

In the last release there was a single set of output files `details.dat` and `summary.dat`. Now there will be one pair of files for each test set in the test file. The names of the files will be of the form `details.x` and `summary.x`, where `x` is 1 for the first test. For successive tests, `x` will range through the single digit integers, then upper case letters, and then the lower case letters. This format was partially chosen because shell commands will operate on files in this order. There is a maximum of 61 test sets in one test file.

## How to Run the New Release

The new release can be run by entering

```
~orin/osiac/sim4
```

However, read below for an easier way.

### New Script Adds Additional Features

I have written a shell script which you can use to simplify running the simulator. This script allows you to specify arbitrary file names for your microinstruction and test files rather than using the fixed names `specs.dat` and `memory.dat`. The script makes a copy of the names you specify into these fixed names, so if your first microinstruction file was named `specs.dat` and you later use a file `specs1.dat`, the original `specs.dat` would be overwritten.

Here is the format of the script:

```
~orin/osiac/dosim4 -options microinstruction_file test_file
```

The three options are `-debug`, `-g`, and `-wcc`. One of the three must appear.

The `debug` option is the original version of the program and it generates a full set of `details.dat` and `summary.dat` files. All registers are shown.

The `g` option is the one I use for grading. It suppress printing the `details.dat` files and it only prints general purpose registers and condition code values. The other registers don't have a defined value at the end of instructions.

The `wcc` option is a version of the grading option that prints output without the condition codes. In some cases, I have used this option to see if an instruction works while ignoring whether the condition codes are right.

The `microinstruction_file` is the file with the format previously described for `specs.dat`. This field must be present.

The `test_file` is the file with the format described for `memory.dat`. If this last field is omitted, the script assumes you already have a `memory.dat` file.

Because repeated use of this script can generate a large number of files, the script will prompt you to see if it should remove all `summary.*` and `details.*` files from previous runs. However, even if you say “n” the new files will overwrite the old ones.

You can make a copy of this script into your own account with the command

```
cp ~orin/osiac/dosim4 .
```

Then you don't have to type the tilde and my name each time. Of course, if I make corrections to the script, you won't be running the latest version. You can even change it if you want to modify features. For example, if you always want to remove old versions, take out the prompt.

Mail any reports of problems to me at address `klein@ece.osu.edu`.