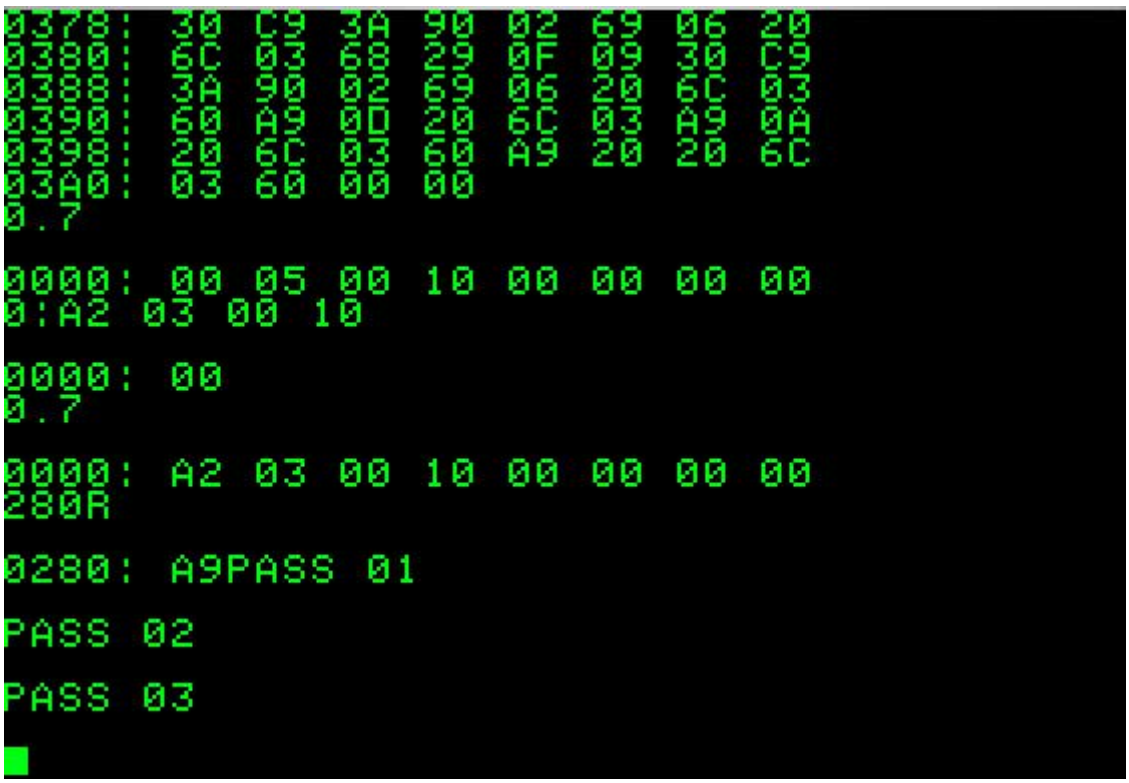

6502 Memory Test

I wrote this basic memory test in order to help kit builders test their boards. The original source code was taken from the 6502 memory test from this compute magazine article.
http://www.atarimagazines.com/compute/issue1/UNIVERSAL_6502_MEMORY_TEST.php

However the this version does not have much in common with the original article. That version took 38 seconds to make a pass on 1K of memory. This version takes under 2 seconds for the same 1k memory block. One issue with that program is if there is an error, the value is read from memory again for display. If the problem is an intermittant read, the displayed value may not be the same as the errored value that was originally read.

In order to run this test. you will need to get this code into memory. You can either type it in or use a terminal program that has a send file function and a serial port. This can be either a superserial card on an Apple II or one of my PS/2-RS232 to keyboard adapters on either an Apple 1 or Apple 2. Keep in mind, that enough of the memory system has to be running in order to enter monitor commands and load the program on either system. The source code is provided, in case you want to modify it or run on a different 6502 based system. Note that if you have an Apple II with the Programmers AID prom, it has a very capable RAM test that can be run directly out of PROM.

Tests

Each pass of this test consists of 6 basic tests.

Test Number	Description
0	All zeros - each byte of memory is verified that a 0x00 value can be written and read
1	All ones - each byte of memory is verified that a 0xff value can be written and read
2	Floating ones - eight passes, starting with 0x01 and moving the 1 bit to left each succeeding pass - 0x02 0x04 0x08 0x10 0x20 0x40 0x80

3	Floating zeros - eight passes, starting with 0x7f and shifting the 0 bit to the right each succeeding pass - 0xBF 0xDF 0xEF 0xF7 0xFB 0xFD 0xFE
4	Address in Address 1 - one pass with low eight bits of the locations address is written to that location - if this fails, you have a problem with one of the low eight address lines (this is pretty unlikely to fail, since you need these address lines in order to load and run this program)
5	Address in Address 2 - one pass with high eight bits of the locations address written to that location - if this fails, you have a problem with one of the eight high address lines

Limitations

The code cannot test the memory in which it runs. Also, the first 8 bytes of zero page are used and because the test is dependant upon the monitor to log results, certain other zero page locations are off limits in the Apple 2. You can always modify the source code in order to relocate it. You can test screen memory, pages 4-7, on the apple2 and keyboard input buffers in page 2, but keep in mind that the monitor is likely to overwrite these locations, once the program returns to the monitor to report an error. You can also test page 1, the stack area, but data may be altered after the test, as the stack is used to print the results.

System	Program Uses	Zero Page in Use
Apple 1	280.3A1	0.8
Apple 2	800.923	0.38

Setup

Using the monitor, enter the starting address and ending address+1 into locations 0-3 of memory.
Example: To test memory from E000 to EFFF on an Apple 1 enter the following into the monitor:
0: 00 E0 00 F0

Run The Test

System	Command
Apple 1	280R
Apple 2	800G

Results

After each pass of the test, "PASS XX" is displayed on the terminal and the test will start over. Press reset to stop the test.
The test will stop and drop into the monitor if it detects an error. The following is displayed

test address expected actual

Here is an example where bit 5 of location E000 was written as a 1 but read as a 0 in the all ones test.
01 E000 FF DF

Files to Download

Source was compiled with a modified version of the ACME85 cross assembler on OS/X. The modifications involved putting the output directly into Apple monitor format, so the output can be directly written into the Apple monitor environment. At some point I'll see if I can get changes posted.

Source file
Apple 1 Command File
Apple 2 Command File

The following in an alternate version. The preceeding version uses the ROR instruction that is not functional on very early 6502 microprocessors. That version will hang after running the first 3 subtests on those very early processors. The following version is slightly altered to avoid using the ROR instruction, so should run on any 6502.

Source File
Apple 1 Command File

Back to Mike's Hobby Home Page