NASA Technical Memorandum 82874

A Generalized Memory Test Algorithm

Edward J. Milner
Lewis Research Center
Cleveland, Ohio

July 1982

NASA

A GENERALIZED MEMORY TEST ALGORITHM

Edward J. Milner

National Aeronautics and Space Administration
Lewis Research Center
Cleveland, Ohio 44135

SUMMARY

Presented in this report is a general algorithm for testing digital computer memory. The test is complete insofar as it checks that 1) every bit of each word can be cleared and set, and 2) bits are not erroneously cleared and/or set elsewhere in memory at the same time. The algorithm is general insofar as it can be applied to any size memory block and any size memory word. It is also concise and efficient, requiring few cycles through memory. Fewer than 400 cycles through memory are required for a test of 16-bit-word memory. The algorithm has been used on a microcomputer having a cycle time of 133 nanoseconds. The memory test took approximately 15 seconds to verify the microcomputer's 32K-by-16-bit memory.

E-1250

INTRODUCTION

A chief requisite of a digital computer is its ability to accurately store information in memory. However, its ability to do so can be hindered for a variety of reasons. For example, improper machine electric field interactions, wiring errors, or shorts may cause memory bits to be cleared or set improperly. An error may occur in the memory word being accessed or it may extend to some other memory word as well. The execution time required for a diagnostic program to check all possible word pattern combinations for the entire memory is unrealistic, even for the fastest computers available. What is needed is a suitable memory test which executes in a reasonable amount of time. A complete memory test at the single-bit-interaction level would be an essential part of such a memory diagnostic.

A diagnostic program was required for memory checkout of a high-speed research microcomputer being designed and developed at the Lewis Research Center. That microcomputer, designated as the DSC-1 (Digital Simulation Computer), was designed to be the computing element of a real-time digital simulator (ref. 1) being developed at Lewis. The DSC-1 was designed to have a cycle time of 133 nanoseconds. Its 32K of 16-bit-word memory was fashioned with provision for expanding the memory to 64K words. By operating up to ten DSC-1's in parallel, the simulator could provide the speed required for real-time simulation of jet engines. This approach to real-time engine simulation offers a number of advantages over current approaches which use hybrid (analog-digital) computers (ref. 2) or mainframe digital computers. The advantages include lower cost, easier programming, more repeatable results, expandability of the hardware, and portability of simulation hardware and software.

It was hoped that an appropriate diagnostic test to check the DSC-1 memory would be available in the literature since many manufacturers have designed computers with this type of memory. However, a search of the literature failed to uncover an algorithm which satisfied our criterion for an acceptable memory test.

Our criterion for an acceptable memory test required that two essential conditions be satisfied. To qualify, a memory diagnostic algorithm must:

1. check that every bit can be cleared and set in each memory location; and
2. check that bits are not erroneously cleared and/or set elsewhere in memory at the same time.

Existing diagnostic algorithms appear to concentrate on condition (1) (ref. 3 to 7) but neglect condition (2). However, condition (1) can be satisfied while serious memory defects still exist. None of these tests adequately address condition (2). Only by addressing this condition in sufficient detail can one be assured that memory is not mistakenly being destroyed. Of the memory test algorithms examined, that presented in ref. 6 most nearly satisfies our two requirements for an acceptable memory test. However, it does not initialize memory and read each memory location before writing into it. Hence, the test fails to detect an important kind of overwrite error. (This shortcoming will be examined in detail later in RESULTS AND DISCUSSION.)

It was necessary, therefore, to develop a memory test algorithm satisfying conditions (1) and (2) above. That new memory test algorithm is documented in this report.

A distinctive characteristic makes the memory test algorithm described herein different from any other currently in the literature. This new algorithm initializes memory to be tested and then reads each memory location just prior to writing into it. By doing so, bit errors violating condition (2) can be effectively determined. As an example, fewer than 400 cycles through 16-bit-word memory are required for the test. Consequently, in the case of the DSC-1 microcomputer, the whole test executes in approximately 15 seconds. Moreover, the algorithm is generalized insofar as it can be applied to any size memory block and any size memory word.

The memory test will be examined in detail in the following sections. Following a description of the algorithm, the basis for the algorithm will be presented and discussed.

ALGORITHM DESCRIPTION

The procedure used by the algorithm is relatively simple. Briefly, it can be summarized as follows. Memory to be tested is first initialized. Next, each memory location is read and then filled with a prescribed binary bit pattern. After the memory-filling process is completed, memory is again read to check that the contents of each location are correct. The entire procedure is then repeated, but this time memory is filled with a different required set of binary bit patterns. The process continues until all required binary bit patterns have been used.

Using this brief overview as a foundation, the algorithm will now be examined in more detail.

The first step in the memory test algorithm is to clear the memory block to be tested. Once cleared, filling the memory block with the first set of

binary bit patterns can take place. The filling process is carried out in the following manner. The first memory word is read to make sure that it is cleared. If it is not cleared, the error is flagged. Otherwise, the first memory word is then loaded with the binary bit pattern consisting of all bits cleared except for the rightmost bit, which is set. If n-bit-word memory is being tested, for example, the pattern

is loaded into the first memory word. Next, the second memory word is read to make sure that it is cleared. Again, if it is not cleared, the error is flagged. Otherwise, it is loaded with the binary pattern consisting of all bits cleared except for the second rightmost bit, which is set. For n-bit-word memory the pattern

would be loaded into the second memory word. Each successive binary bit pattern is formed by simply rotating the current pattern left one bit. Repeating the process for the third memory word, then, produces the binary bit pattern

The algorithm continues to fill successive memory locations in this fashion until the entire memory block to be tested has been filled. When filled, memory should contain the pattern displayed in figure 1. Notice that the "sliding one" pattern repeats every n memory words. Once memory has been filled, the next step in the algorithm is to check memory by again reading each memory location to verify that it contains the value with which it was filled. If an error is detected, it is flagged. This portion of the test will be referred to as "sliding one-part 1" in following discussions. (For convenience, TABLE I summarizes memory initialization and loading order for each part of the algorithm.)

In the next part of the test, denoted as "sliding one-part 2", the memory is filled in reverse order, as follows. Memory is initialized by clearing each bit in each word as before. The last memory word in the block is then read to make sure that all its bits are cleared. The last binary bit pattern used in "sliding one-part 1" is then loaded into this memory word. Next, the second last memory word is read to make sure that all its bits are cleared. The memory word is filled with the binary bit pattern obtained by rotating right one bit the binary pattern just used to fill the last memory word. The memory-filling process continues in this reverse order with each successive binary bit pattern obtained by rotating the current pattern right one bit. When the memory block has been filled, the first memory word contains the pattern

and the memory block should again contain the pattern displayed in figure 1. As before, once memory has been filled, each memory location is again read to confirm that that it contains the value with which it was filled.

"Sliding one-part 3" and "sliding one-part 4" are essentially the same as parts 1 and 2, respectively. They differ only in that memory is initialized "high" for these portions of the test. Every bit of each word is set, not cleared as before, prior to beginning the filling process. The filling and checking processes are as described above except that each word is read to make sure that all its bits are set before it is filled with a binary bit pattern.

The algorithm requires cycling through all possible initial patterns of bits. That is, for the second pass through "sliding one-part 1," the binary pattern

is used as the initial pattern in the first memory word. When filled, memory should then contain the pattern displayed in figure 2. The algorithm continues cycling until all n possible initial (sliding one) bit patterns have been used. For the nth and final pass, the initial pattern in the first memory word is the binary pattern

Finally, the algorithm repeats the entire procedure using a "sliding zero" in place of the "sliding one." For the first pass through "sliding zero-part 1," the initial pattern loaded into the first memory word is

When filled this first time, memory should contain the pattern displayed in figure 3.

RESULTS AND DISCUSSION

As mentioned before, the distinctive characteristic of the memory test algorithm just described is the reading of each memory word before it is loaded with a particular binary bit pattern. This feature offers a convenient means to determine whether bits have been erroneously cleared and/or set elsewhere within memory from some access to a particular memory location.

Referring to figure 4, suppose that loading some memory location i causes an error of one or more bits to be cleared and/or set elsewhere in memory. The error falls within one, or a combination, of three distinct categories. Namely, CASE I) bits are cleared and/or set erroneously in memory words preceeding location i (area I in fig. 4); CASE II) bits are cleared and/or set erroneously in memory words following location i (area II in fig. 4); and CASE III) bits are cleared and/or set erroneously in location i itself.

Let us now examine each of these cases individually.

CASE I:

A. If bits are cleared in a preceeding location, the error will be detected by "sliding one-part 4" of the algorithm. Recall that

for this portion of the test, memory is initialized with all bits set, and the memory block is filled in reverse order from last word to first word. The error will be discovered when the algorithm attempts to load a binary pattern into a memory word whose value has changed from its initial condition value of all bits set.

B. If bits are set in a preceeding location, the error will be detected by "sliding one-part 2" of the algorithm. Here, an attempt will be made to load a binary pattern into a memory word whose value has changed from its initial condition value of all bits cleared.

CASE II:

A. If bits are cleared in a following location, the error will be detected by "sliding one-part 3" of the algorithm. For this portion of the test, memory is initialized with all bits set and the memory block is filled in a forward direction from first word to last. Again, the value of a memory word will have changed from its initial condition of all bits set.

B. If bits are set in a following location, the error will be detected by "sliding one-part 1" of the algorithm in similar fashion.

CASE III:

If extra bits are set within the word itself, the "sliding one" portion of the algorithm will detect the error. It will do so because after the memory block is filled, each memory location is read to verify that it contains the value with which it was supposed to be loaded. Since a valid "sliding one" pattern has only one bit set, an extra bit set in the word will be discovered immediately. The "sliding-zero" test, however, is required to detect the error of extra bits being cleared within the word itself. This error will be detected while reading memory after it is filled because a valid "sliding zero" pattern has only one bit cleared.

Since erroneous clearing and/or setting of bits elsewhere in memory is detected, the algorithm fulfills the second condition required for an acceptable memory test. It certainly also fulfills condition (1) since each bit of every word in memory is cleared and set not only during the "sliding one" test but also again during the "sliding zero" test.

CONCLUDING REMARKS

Presented in this report is an algorithm for testing digital computer memory. The scheme is general insofar as it can be applied to any size memory block and any size memory word. The algorithm is concise and efficient. Fewer than 400 cycles through memory are required for a test of 16-bit-word memory. Using a microcomputer having a cycle time of 133 nanoseconds, approximately 15 seconds were required to test its 32K-by-16-bit memory. The algorithm is a complete memory test at the single-bit-interaction level. It will also detect basic types of pattern sensitivity

because each memory bit is required to tolerate neighboring bits at both the same and opposite logic levels as itself. However, it was not designed to thoroughly test for word pattern sensitivity because such a test would require too much program execution time to be practical. Likewise, detecting intermittent errors is hardly plausible unless they occur while the diagnostic test is executing.

REFERENCES

1. Blech, Richard A.; and Arpasi, Dale J.: An Approach to Real-Time Simulation Using Parallel Processing. NASA TM 81731, 1981.

2. Szuch, John R.; Seldner, Kurt; and Cwynar, David S.: Development and Verification of Real-Time Hybrid Computer Simulation of F100-PW-100(3) Turbofan Engine. NASA TP 1034, 1977.

3. Lilley, Robert W.: Test Program for 4-K Memory Card, JOLT Microprocessor. (TM-33, Ohio University; NASA Grant NGR-36-009-017.) NASA CR-148770, 1976.

4. Nabers, Steve: 6502 Comprehensive Memory Test Program. Interface Age, Vol. 4, No. 4, Apr. 1979, pp. 140-145.

5. Duncan, Ray: Z-80 Memory Test. Dr. Dobb's Journal (Computer Calisthenics and Orthodontia), No. 52, Feb. 1981, pp. 22-23.

6. Borer, A. J.: Total Memory Test. Microprocessors and Microsystems, Vol. 4, No. 4, May 1980, pp. 141-144.

7. Grappel, Robert D.: M68000 Diagnostic Program Tests Memory. EDN Magazine, Vol. 26, No. 8, April 15, 1981; pp. 157-158.

TABLE I. - SUMMARY OF MEMORY INITIALIZATION AND MEMORY LOAD ORDER

	Memory initialization	Memory load order	Text use reference
Sliding one-part 1	Clear	Forward	Case II-B
Sliding one-part 2	Clear	Reverse	Case I-B
Sliding one-part 3	Set	Forward	Case II-A
Sliding one-part 4	Set	Reverse	Case I-A
Sliding zero-part 1	Clear	Forward	Case III
Sliding zero-part 2	Clear	Reverse	Case III
Sliding zero-part 3	Set	Forward	Case III
Sliding zero-part 4	Set	Reverse	Case III

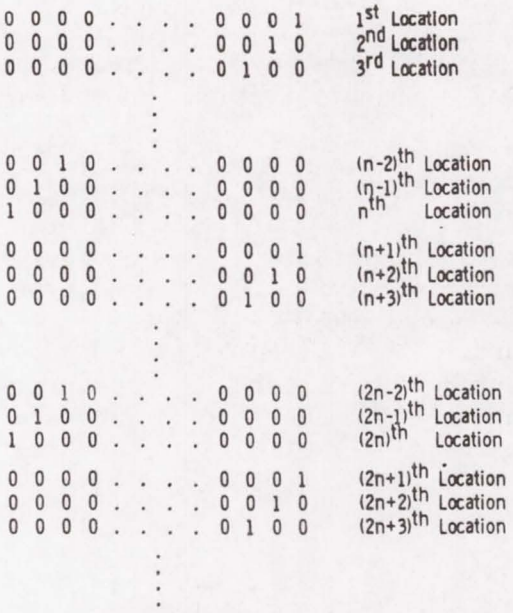

Figure 1. - Memory contents after first pass through "Sliding one" test.

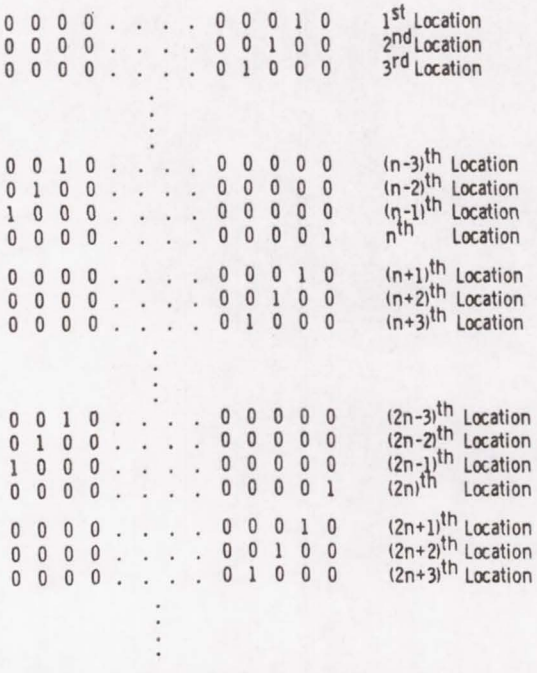

Figure 2. - Memory contents after second pass through "Sliding one" test.

1	1	1	1	.	.	.	.	1	1	1	0	1st Location
1	1	1	1	.	.	.	.	1	1	0	1	2nd Location
1	1	1	1	.	.	.	.	1	0	1	1	3rd Location
1	1	0	1	.	.	.	.	1	1	1	1	(n-2)th Location
1	0	1	1	.	.	.	.	1	1	1	1	(n-1)th Location
0	1	1	1	.	.	.	.	1	1	1	1	nth Location
1	1	1	1	.	.	.	.	1	1	1	0	(n+1)th Location
1	1	1	1	.	.	.	.	1	1	0	1	(n+2)th Location
1	1	1	1	.	.	.	.	1	0	1	1	(n+3)th Location
1	1	0	1	.	.	.	.	1	1	1	1	(2n-2)th Location
1	0	1	1	.	.	.	.	1	1	1	1	(2n-1)th Location
0	1	1	1	.	.	.	.	1	1	1	1	(2n)th Location
1	1	1	1	.	.	.	.	1	1	1	0	(2n+1)th Location
1	1	1	1	.	.	.	.	1	1	0	1	(2n+2)th Location
1	1	1	1	.	.	.	.	1	0	1	1	(2n+3)th Location

Figure 3. - Memory contents after first pass through "Sliding zero" test.

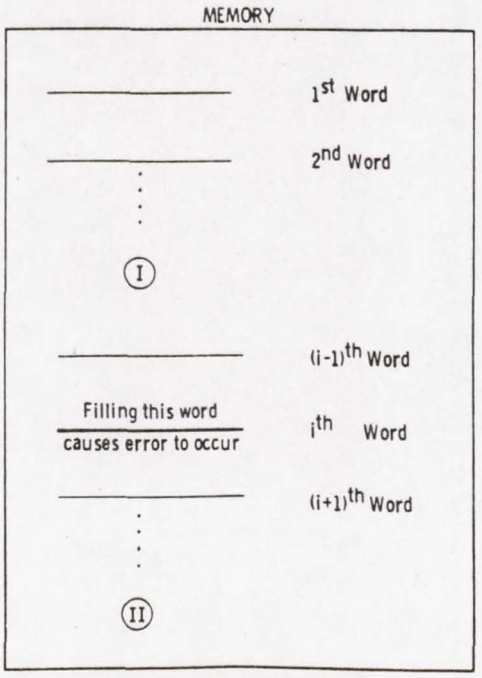

Figure 4. - Memory block being tested. Loading i^{th} word causes error in memory.

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-82874		

4. Title and Subtitle	5. Report Date
A GENERALIZED MEMORY TEST ALGORITHM	July 1982
	6. Performing Organization Code 505-32-6B
7. Author(s) Edward J. Milner	8. Performing Organization Report No. E-1250
	10. Work Unit No.
9. Performing Organization Name and Address National Aeronautics and Space Administration Lewis Research Center Cleveland, Ohio 44135	11. Contract or Grant No.
	13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, D.C. 20546	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

Presented is a general algorithm for testing digital computer memory. The test checks that 1) every bit can be cleared and set in each memory word, and 2) bits are not erroneously cleared and/or set elsewhere in memory at the same time. The algorithm can be applied to any size memory block and any size memory word. It is concise and efficient, requiring very few cycles through memory. For example, a test of 16-bit-word-size memory requires only 384 cycles through memory. Approximately 15 seconds were required to test a 32K block of such memory, using a microcomputer having a cycle time of 133 nanoseconds.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
Computer memory test; Memory diagnostic; General memory test; Memory error test; Memory failure test; Computer memory diagnostic	Unclassified - unlimited STAR Category 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price*
Unclassified	Unclassified		

* For sale by the National Technical Information Service, Springfield, Virginia 22161