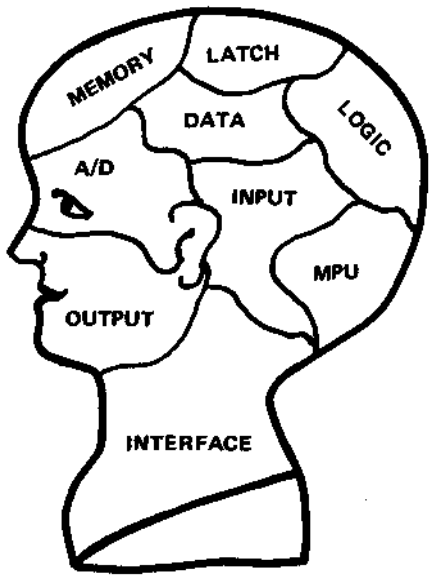
Questdata
Volume 1 Issue #11 ©

DOWN MEMORY LANE

YOUR computer will say, "thanks for the good memories" after running this program. The program, as written, will quickly test all 4K of a Super Elf and report any failed locations. The comprehensive memory test fits snugly into the 256 bytes of RAM of the original Elf (which moves up to locations 9800–98FF when the original Elf is plugged into the Super Expansion System.

After loading the program at locations 9800–9872, you can enter a long jump at location 0000 (C0 98 00) or you can use a monitor jump to location 9800. If your computer fails the test, what do you do? By depressing the INPUT key twice (once for hi addr. and once for low addr.) you find the location or block of memory which has bad memory. The Quest Super Elf Expansion Board uses 2102 static RAM's. The memory of a single 2102 contains 1024 locations (computer people call it 1K although it is obviously not an even thousand). The 2102 memory chip contains one bit of DATA in each of the 1K locations.

You now know which 1K block in which the memory failure occurred. If the location after INPUT is pressed twice is revealed to be 013F, you have a memory problem in the first 0000–03FF block (1K). This block is located in the back or farthest away position as you look at the board with the parallel ports nearest you and the board number and Quest logo so you can read it. If you wanted you could replace the 8-2102's with new memory chips, and the problem would be solved. With memory of the 2102 type being about $1.00 and everyone wanting to save money, we proceed to finding the exact culprit. To discover which 2102 has devilishly foiled our otherwise perfect memory we decode the failed bits. Depressing the INPUT key reveals that the information inside (DATA- D0 thru D7) should have been 02 but the actual information stored was 00. In other words, 0000 0000 should have been 0000 0010. "Ha, ha–got you!" you exclaim. You then proceed to replace the memory chip U17 (the second from the left in 1K memory bank). You restart the memory program and all the memory checks out OK. Its actually kind of fun detective work to track down the murdered chip and discover who-done-it.



The memory array on the Super Expansion is organized as follows:

TOP ROW is 0000-03FF
2nd ROW is 0400-07FF
3rd ROW is 0800-0BFF
4th ROW is 0C00-0FFF

Bit 0 is on the LEFT with Bit 7 on the RIGHT.

You deserve some explanation of the Assembler notations used in the program. One of the products on the drawing board at Quest is an Assembler, so that this program will be a good "how to" reference for you in the future. Since Assemblers yield insight into machine language programming, the time to start thinking about them is at hand.

ASSEMBLER EXPLAINED

So far we have been "assembling programs by hand," this process, given the regularity of the COSMAC code and the small size of our programs, is is not too bad. Probably, however, you have found

yourself wishing that you had something the equivalent of relative addressing. The Assembler gives you the ability to jump to **LABELS** instead of absolute value hex locations. After you construct a program using the mnemonics, symbols and labels, you make an Assembler run and out comes hex code and memory locations. Historically, Assemblers were one of the first higher level languages. They make it easier for programmers to write, document and maintain software.

The first thing you notice about the Assembler listing is that there are both line numbers and locations. The line numbers are used to reference specific lines of code, so that a programmer knows exactly what is happening. An error in line 18 would tell the programmer that perhaps there is something wrong with the symbol he is using. It is the same reason that books have page numbers -- "please turn to page 53, for today's grammer lesson," says the English teacher.

"Rule number one is that all comments and remarks have two **PERIODS** . . in front of them," says the mystic rites of Assemblers teacher. The two periods in front of the remarks is the RCA convention and it makes it easy to draw neat little boxes, so it is an OK rule. The rest of the line after the two periods is ignored by the Assembler but the diligent programmer should not ignore putting them in. The rule of thumb is that it is far better to have too many comments and remarks (they can always be ignored) than too few. Programmers who ignore this are savegly sacrificed upon the alter of GIGO (Garbage In, Garbage Out). Judging from the level of documentation submitted with your programs to QUESTDATA, you need not fear being sacrificed to the god GIGO.

Let's look at line 18. START=#0000 is called a directive by RCA (probably to avoid calling it a pseudo-op -- a word **IBM** has copyrighted). Anyhow, the **EQUATE (= sign)** directs the Assembler to identify the symbol START with 0000. These two become twins or two peas in a pod – when the Assembler sees the word START it thinks to itself, "Ah, me put'em in 0000, now." The number 0000 prefixed by the "#" sign is a constant and is one of several constants used in the program. Here it is a hex constant as indicated by the "#"symbol. Register constants have two hex digits in this program. The stack register is hex 02 (referencing one of the 16 scratchpad registers).

All the assignments have been made and the program is ready to begin at line 162. ORG is the RCA directive to start assigning locations from that location on with the information given on the ORG statement. ORG MEMTST has been assigned the location value 9800 on line 20. So putting it all together the Assembler shorthand is saying, "start assigning locations at location 9800." Without an ORG statement the Assembler would assume that you were starting the program at 0000 (which is OK if you are starting at 0000). ORG #9800 would have been another choice for starting the program at location 9800 hex. ORG's can be used anytime. One of the really nice things about an Assembler is that it gives you this freedom of choice. You can represent things in a manner which is convenient, meaningful and easy to use (once you have learned its tricks).

Line number 190 contains the label SUBRET:. The **COLON** : is used to the right of the symbol SUBRET to identify that this is a "statement label"' The computer notes the location of all labels in its "symbol table" and notes both the label and the loc. to which it refers. Now when the programmer wants to jumb to the SUBRET location (location 9814). all that is needed is BR SUBRET. This is exactly what happens in line 201, where the programmer uses this jump to the SUBRET label. Since the computer knows the exact hex location of SUBRET it is possible to refer to location 9816 as SUBRET+2. The equate at line 21 (ADDBUF=#981E) is used by the Assembler's computation ability later on in the program. The computer knows (computers aren't so dumb) that ADDBUF=#981E from line 21. So when it comes to line 228 and sees ADDBUF+1, it goes to its table and finds ADDBUF and adds one, thus arriving at 1F for the lower byte of this address (1E + 1 = 1F).

The **ASTERISK** * in the RCA Assembler is a handy device to refer to the location the computer is now at. For example, when we have wanted to wait for an INPUT button to be pressed we have used 3F 18 (18 being the address of the 3F), now we do not even have to glance at where we are on the location part of the coding sheet. BN4 * will branch back to the start of the two byte BN4 instruction. An ORG*+ 6 will skip six locations from the point it is found in the program.

Anything inside a **PARENTHESIS** ( ) is considered to be an address by the Assembler. This represents another kind of constant, an "address constant." An example of this is line 170 in our MEMTST program. LDI A.1 (ADDBUF) will cause the Assembler to frantically scramble for its symbol table (ADDBUF= #981E), and put the high part of its findings (the A.1 or 98) into the second half of the F8 load into the D part of the load immediate. In the case of , A(START) it takes the whole address. Line number 205 loads 0000 into the two bytes starting with 981E. The comma in front of the ,A(START) is another Assembler construction. It tells the Assembler to obtain the address, or address part, and insert it directly into the program at that point.

The MEMTST program, like all programs, uses the **END** directive to halt the Assembly process.

Assemblers offer many different ways to declare data, specify addresses and make the programmer's life easier. While Assemblers are not difficult, they do take some study to understand. Assemblers are really helpful in putting together MACHINE LANGUAGE with half the effort.

```
0000     1 ..
0000     2 ..MEMORY TEST PROGRAM
0000     3 ..
0000     4 ..     VERSION 1.0
0000     5 ..
0000     6 ..TO LOAD AND RUN PROGRAM:
0003     7 ..
0000     8 ..  LOAD PROGRAM AT LOCATION DESIGNATED BY THE
0000     9 ..  SYMBOL "MEMTST". USE SUPER MONITOR OR OTHER
0000    10 ..  MEANS TO BRANCH INTO THE PROGRAM AT LOCA-
0000    11 ..  TION 00. **** CAUTION **** DO NOT USE REGS
0000    12 ..  2,4,5, OR 6 AS PROGRAM COUNTER WHEN ENTER-
0000    13 ..  THIS PROGRAM.
0000    14 ..
0000    15 ..OPERATION OF PROGRAM:
0000    16 ..
0000    17 ..1. PROGRAM PARAMETERS:
0000    18          START=#0000
0000    19           STOP=#0FFF
0000    20         MEMTST=#9800
0000    21         ADDBUF=#981E
0000    22          STACK=#98FF
0000    23          EXADR=#8000
0000    24 ..
0000    25 ..  THE START ADDRESS FOR THE PORTION OF MEMORY
0000    26 ..  TO BE TESTED IS DEFINED BY THE PARA. "START"
0000    27 ..  IT IS LOCATED AT THE ADDRESS DEFINED BY THE
0000    28 ..  PARAMETER "ADDBUF". YOU CAN PATCH A NEW ONE
0000    29 ..  AT ADDBUF+0 AND ADDBUF+1. SIMILARLY, THE
0000    30 ..  STOP ADDRESS FOR THE MEMORY UNDER TEST IS
0000    31 ..  DEFINED BY THE PARA. "STOP". YOU CAN PATCH
0000    32 ..  A NEW STOP ADDRESS AT ADDBUF+2 AND ADDBUF+3
0000    33 ..
0000    34 ..  THE LOCATION OF THE ADDRESS BUFFER CAN BE
0000    35 ..  PATCHED TO A DIFFERENT LOCATION IN MEMORY
0000    36 ..  BY CHANGING MEMTST+1 TO THE NEW HI ADDRESS
0000    37 ..  AND MEMTST+39(DEC) TO THE NEW LO ADDRESS.
0000    38 ..
0000    39 ..  THE STACK ADDRESS IS DEFINED BY "STACK". A
0000    40 ..  NEW STACK AREA CAN BE PATCHED IN BY CHANG-
0000    41 ..  ING MEMTST+4 TO THE NEW HI ADDRESS, AND BY
0000    42 ..  CHANGING MEMTST+7 TO THE NEW LO ADDRESS.
0000    43 ..
0000    44 ..  THE EXIT ADDRESS CAN BE PATCHED TO A DIFF-
0000    45 ..  ERENT QUARTER-K OF MEMORY THAN CURRENTLY
0000    46 ..  DEFINED BY "EXADR" (BUT NOT TO ANY ADDRESS
0000    47 ..  ON THE QUARTER-K THAN 00) BY CHANGING THE
0000    48 ..  BYTE AT MEMTST+110(DEC).
0000    49 ..
0000    50 ..  THE PROGRAM IS RELOCATABLE PROVIDED THE
0000    51 ..  START/STOP ADDRESS BUFFER POINTER (R4) IS
0000    52 ..  PROPERLY HANDLED AS SUGGESTED ABOVE, THE
0000    53 ..  STACK IS CARED FOR, AND A FURTHER PATCH
0000    54 ..  MADE AT MEMTST+11(DEC): REPLACE THE DATA
0000    55 ..  THERE WITH THE DESIRED HI ADDRESS OF THE
0000    56 ..  NEW QUARTER-K.
0000    57 ..
0000    58 ..2. PROGRAM OPERATION:
0000    59 ..
0000    60 ..  ONCE THE PROGRAM IS ENTERED, IT STARTS
0000    61 ..  LOADING TEST PATTERNS IN BLOCKS AND COMP-
0000    62 ..  ARES MEMORY IMMEDIATELY AFTER EACH BLOCK IS
0000    63 ..  LOADED. ALL VALUES OF 00 THROUGH FF ARE PUT
0000    64 ..  INTO EACH CELL OF MEMORY IN THE REGION DE-
0000    65 ..  SIGNATED. THE TECHNIQUE USED IS TO START AT
0000    66 ..  THE TAIL-END OF MEMORY FIRST WITH 00 AND
0000    67 ..  USE THE NEXT HIGHER COUNT OF THE PATTERN IN
0000    68 ..  THE PRIOR ADDRESS ETC. UNTIL THE DESIGNATED
0000    69 ..  BLOCK OF MEMORY HAS BEEN FILLED. THE RE-
0000    70 ..  SULTING SET OF PATTERNS IS THEN CHECKED TO
0000    71 ..  DETERMINE IF ANY ERRORS HAVE OCCURED IN THE
0000    72 ..  STORAGE OPERATION. ERRORS FOUND ARE REPORT-
0000    73 ..  ED TO THE USER AS EXPLAINED BELOW. THEN THE
0000    74 ..  PROGRAM RETURNS TO THE LAST LOCATION OF THE
0000    75 ..  BLOCK OF MEMORY, STORES 01 AND REPEATS THE
0000    76 ..  ABOVE STORE AND COMPARE PROCEDURE. THIS RE-
0000    77 ..  PEATS OVER AND OVER UNTIL FF HAS BEEN STOR-
0000    78 ..  ED AND COMPARED IN THE LAST LOCATION. THEN
0000    79 ..  THE PROGRAM LOADS EXADR IN R0, ASSIGNS R0
0000    80 ..  TO X AND SEPS TO R0.
0000    81 ..
0000    82 ..  IF AN ERROR IS DETECTED, THE FOLLOWING
0000    83 ..  PROCEDURE IS FOLLOWED:
0000    84 ..  A.DEPRESS I KEY, THE HI ADDRESS OF THE
0000    85 ..    FAILED MEMORY LOCATION IS DISPLAYED ON
0000    86 ..    HEX DISPLAY.
0000    87 ..  B.DEPRESS I KEY, THEN THE LO ADDRESS OF THE
0000    88 ..    FAILED LOCATION IS DISPLAYED.
0000    89 ..  C.DEPRESS I KEY, THEN THE CURRENT TEST DATA
0000    90 ..    ("SHOULD-BE" VALUE) IS DISPLAYED.
0000    91 ..  D.DEPRESS I KEY, THEN THE FAILED VALUE IS
0000    92 ..    DISPLAYED.
0000    93 ..  A SUBSEQUENT I KEY DEPRESSION WILL RESUME
0000    94 ..  THE TESTING OF MEMORY.
0000    95 ..
0000    96 ..  WHILE THE PROGRAM IS RUNNING AND NO ERRORS
0000    97 ..  HAVE BEEN DETECTED, THE CURRENT LAST LOCA-
0000    98 ..  TION TEST VALUE IS DISPLAYED. THE INDICAT-
0000    99 ..  ION OF AN ERROR IS THE STEADY DISPLAY OF EE
0000   100 ..  AND THE GLOWING OF THE Q LED.
0000   101 ..
0000   102 ..3.TERMINATION OF RUN
0000   103 ..
0000   104 ..  TERMINATION OF THE PROGRAM IS INDICATED BY
0000   105 ..  A STEADY 00 ON THE HEX DISPLAY. IF YOU THEN
0000   106 ..  DEPRESS THE I KEY, THE PROGRAM WILL RETURN
0000   107 ..  TO THE SUPER MONITOR.
0000   108 ..
```

```
0000      109 ..REGISTER ASSIGNMENTS:
0000      110 ..REG.
0000      111 .. 0   USED FOR EXIT FROM PROGRAM. LOADED WITH
0000      112 ..     "EXADR" AND SEP'D, AS WELL AS SEX'D.
0000      113         EXITR=#00
0000      114 .. 1   -
0000      115 .. 2   USED AS STACK POINTER
0000      116            ST=#02
0000      117 .. 3   -
0000      118 .. 4   USED FOR POINTER TO START AND STOP ADD-
0000      119 ..     RESS BUFFER. THE BUFFER ADDRESS IS DES-
0000      120 ..     IGNATED BY "ADDBUF"
0000      121        ADBPTR=#04
0000      122 .. 5   USED FOR THE SUBROUTINE
0000      123           SUB=#05
0000      124 .. 6   PROGRAM COUNTER
0000      125            PC=#06
0000      126 .. 7.0 DUAL USE:
0000      127 ..     1. HOLDS MAJOR LOOP TEST PATTERN
0000      128 ..     2. PROVIDES MAJOR LOOP CONTROL
0000      129        TSTPAT=#07
0000      130 .. 7.1 USED TO STORE A FAILED VALUE DETECTED
0000      131 ..     (IF ONE IS FOUND).THE FORM OF THE DATA
0000      132 ..     WILL THEN BE THE XOR OF THE CURRENT
0000      133 ..     PATTERN.
0000      134        FALVAL=#07
0000      135 .. 8   HOLDS THE CURRENT TEST PATTERN IN EITH-
0000      136 ..     ER THE STORE OR COMPARE LOOPS.
0000      137        CURPAT=#08
0000      138 .. 9   HOLDS THE CURRENT ADDRESS OF THE MEMORY
0000      139 ..     BEING TESTED. INITIALIZED AT THE STOP
0000      140 ..     ADDRESS AND INDEXED BACKWARDS FOR THE
0000      141 ..     PATTERN STORE, THEN INDEXED FORWARDS FOR
0000      142 ..     THE COMPARE
0000      143        MEMPTR=#09
0000      144 .. A   NUMBER OF BYTES COUNTER FOR USE IN THE
0000      145 ..     MEMORY STORE LOOP
0000      146        STBYCT=#0A
0000      147 .. B   NUMBER OF BYTES COUNTER FOR USE IN THE
0000      148 ..     TEST COMPARE LOOP
0000      149        CMBYCT=#0B
0000      150 ..     NOTE: REGS A AND B INITIALLY HOLD IN THE
0000      151 ..     MAJOR LOOP THE VALUE OF THE NUMBER OF
0000      152 ..     BYTES OF THE BLOCK OF MEMORY UNDER TEST
0000      153 ..     PLUS AN ADDITIVE FACTOR OF 00FF(HEX), TO
0000      154 ..     ALLOW DECREMENTING AND TESTING MORE EFF-
0000      155 ..     ICIENTLY.
0000      156 .. C   -
0000      157 .. D   -
0000      158 .. E   -
0000      159 .. F   -
0000      160 ..
0000      161 ..
9800      162        ORG MEMTST
```

```
9800      163 ..
9800      164 .....
9800      165 ..
9800      166 ..INITIALIZE THE PROGRAM
9800      167 ..
9800      168 .....
9800      169 ..
9800 F898 170        LDI A.1(ADDBUF)      ..SET THE HI ADDRESS
9802 B4   171        PHI ADBPTR           ..OF THE START/STOP
9803      172                             ..ADDRESS BUFFER
9803      173 ..
9803 F898 174        LDI A.1(STACK)       ..SET THE STACK
9805 B2   175        PHI ST               ..POINTER
9806 F8FF 176        LDI STACK
9808 A2   177        PLO ST
9809 E2   178        SEX ST
980A      179 ..
980A F898 180        LDI A.1(*)           ..SET THE HI ADDRESS
980C B6   181        PHI PC               ..OF PROGRAM COUNTER
980D B5   182        PHI SUB              ..AND SUBROUTINE
980E      183                             ..POINTER
980E      184 ..
980E F815 185        LDI SUBENT           ..SET THE LO ADDRESS
9810 A5   186        PLO SUB              ..OF THE SUBROUTINE
9811      187 ..
9811 F822 188        LDI MAIN             ..SET THE LO ADDRESS
9813 A6   189        PLO PC               ..AND SEP THE PROG
9814 D6   190 SUBRET:SEP PC               ..COUNTER (THIS IN-
9815      191                             ..STRUCTION PROVIDES
9815      192                             ..FOR THE RETURN FOR
9815      193                             ..THE SUB ALSO)
9815      194 ..
9815 52   195 SUBENT:STR ST               ..SUBROUTINE ENTRY
9816 64   196        OUT 4                ..DISPLAY THE DATA
9817 22   197        DEC ST
9818      198 ..
9818 3F18 199        BN4 *                ..WAIT FOR THE I KEY
981A 371A 200        B4 *
981C 3014 201        BR SUBRET
981E      202 ..
981E      203 ..ADDRESS BUFFER
981E      204 ..
981E 0000 205        .A(START)
9820 0FFF 206        .A(STOP)
9822      207 ..
9822      208 .....
9822      209 ..
9822      210 ..START THE PROGRAM
9822      211 ..
9822      212 .....
9822      213 ..
9822 F800 214 MAIN:  LDI 0                ..SET THE INITIAL TEST
9824 A7   215        PLO TSTPAT           ..PATTERN
9825      216 ..
```

```
9825          217 ..........................................
9825          218 ..
9825          219 ..MAJOR LOOP
9825          220 ..
9825          221 ..........................................
9825          222 ..
9825 A8       223 PATST: PLO CURPAT        ..STORE AND DISPLAY
9826 52       224        STR ST            ..THE CURRENT TEST
9827 64       225        OUT 4             ..PATTERN
9828 22       226        DEC ST
9829          227 ..
9829 F81F     228        LDI ADDBUF+1      ..COMPUTE THE NUMBER
982B A4       229        PLO ADBPTR        ..OF BYTES OF THE TEST
982C 44       230        LDA ADBPTR        ..MEMORY.  SET THE
982D 52       231        STR ST            ..BYTE COUNTERS AND
982E 14       232        INC ADBPTR        ..THE ADDRESS POINTER
982F 04       233        LDN ADBPTR
9830 A9       234        PLO MEMPTR
9831 F7       235        SM
9832 AA       236        PLO STBYCT
9833 AB       237        PLO CMBYCT
9834 24       238        DEC ADBPTR
9835 24       239        DEC ADBPTR
9836 24       240        DEC ADBPTR
9837 44       241        LDA ADBPTR
9838 52       242        STR ST
9839 14       243        INC ADBPTR
983A 04       244        LDN ADBPTR
983B B9       245        PHI MEMPTR
983C 77       246        SMB
983D FC01     247        ADI 1
983F BA       248        PHI STBYCT
9840 BB       249        PHI CMBYCT
9841          250 ..
9841 E9       251        SEX MEMPTR        ..SET THE MEMORY POIN-
9842          252                          ..TER AS THE X REGIST-
9842          253                          ..ER SO THE XOR COMM-
9842          254                          ..AND CAN BE USED FOR
9842          255                          ..THE TEST OF MEMORY
9842          256 ..
9842 18       257        INC CURPAT        ..ADJUST FOR THE LOGIC
9843          258 ..
9843          259 ..........................................
9843          260 ..
9843          261 .. TEST PATTERN STORE LOOP
9843          262 ..
9843          263 ..........................................
9843          264 ..
9843 28       265 STOR:  DEC CURPAT        ..STORE THE CURRENT
9844 88       266        GLO CURPAT        ..PATTERN IN THE MEM-
9845 73       267        STXD              ..ORY BLOCK UNDER TEST
9846 2A       268        DEC STBYCT
9847 9A       269        GHI STBYCT
9848 3A43     270        BNZ STOR
```

```
984A          271 ..
984A 19       272        INC MEMPTR        ..RE-POINT TEST MEMORY
984B          273                          ..POINTER TO LAST LOC-
984B          274                          ..ATION STORED
984B          275 ..
984B          276 ..........................................
984B          277 ..
984B          278 .. TEST COMPARE LOOP
984B          279 ..
984B          280 ..........................................
984B          281 ..
984B 88       282 COMP:  GLO CURPAT        ..COMPARE MEMORY
984C F3       283        XOR               ..AGAINST THE SET OF
984D 3260     284        BZ GTEST          ..PATTERNS JUST STOR-
984F          285                          ..ED
984F          286 ..
984F          287 ..........................................
984F          288 ..
984F          289 .. FAILURE HANDLING
984F          290 ..
984F          291 ..........................................
984F          292 ..
984F 7B       293        SEQ               ..TURN ON THE Q LED,
9850 B7       294        PHI FALVAL        ..DISPLAY THE EE ERROR
9851 E2       295        SEX ST            ..INDICATION ON THE
9852 F8EE     296        LDI #EE           ..HEX DISPLAY.  THEN
9854 D5       297        SEP SUB           ..UPON USER DEPRESSION
9855 99       298        GHI MEMPTR        ..OF THE I KEY PROVIDE
9856 D5       299        SEP SUB           ..AT EACH DEPRESSION,
9857 89       300        GLO MEMPTR        ..THE FOLLOWING DATA:
9858 D5       301        SEP SUB           ..THE HI ADDRESS OF
9859 88       302        GLO CURPAT        ..THE FAILED MEMORY,
985A D5       303        SEP SUB           ..THE LO ADDRESS, THE
985B 97       304        GHI FALVAL        ..EXPECTED PATTERN AND
985C F3       305        XOR               ..THE ACTUAL PATTERN
985D D5       306        SEP SUB           ..FOUND.  AT A SUBSE-
985E E9       307        SEX MEMPTR        ..QUENT DEPRESSION OF
985F 7A       308        REQ               ..THE I KEY, CLEAR THE
9860          309                          ..ERROR INDICATION AND
9860          310                          ..GO ON
9860          311 ..
9860          312 ..........................................
9860          313 ..
9860          314 .. (RESUME COMPARE LOOP)
9860          315 ..
9860          316 ..........................................
9860 18       317 GTEST: INC CURPAT        ..SET THE NEXT PATTERN
9861 19       318        INC MEMPTR        ..CHECK FOR END OF
9862 28       319        DEC CMBYCT        ..THIS LOOP.  IF NOT
9863 98       320        GHI CMBYCT        ..DONE, DO THE COMPARE
9864 3A4B     321        BNZ COMP
9866          322 ..
9866          323 ..........................................
9866          324 ..
```

```
9866                    325 .. END OF THE MAJOR LOOP-THROUGH
9866                    326 ..
9866                    327 ..........................................
9866                    328 ..
9866 E2                 329         SEX ST              ..SET THE NEXT PATTERN
9867 17                 330         INC TSTPAT          ..AND CHECK FOR COM-
9868 87                 331         GLO TSTPAT          ..LETION. IF NOT DONE,
9869 3A25               332         BNZ PATST           ..START STORING THE
986B                    333                             ..NEXT PATTERN
986B                    334 ..
986B                    335 ..........................................
986B                    336 ..
986B                    337 .. DONE WITH JOB
986B                    338 ..
986B                    339 ..........................................
986B                    340 ..
986B D5                 341         SEP SUB             ..DISPLAY THE END IN-
986C A0                 342         PLO EXITR           ..DICATION AND JUMP TO
986D F800               343         LDI A.1(EXADR)      ..THE EXIT LOCATION
986F B0                 344         PHI EXITR
9870 E0                 345         SEX EXITR
9871 D0                 346         SEP EXITR
9872                    347         END
```

LISTING

```
9800  F898 B4F8 9802 F8FF A2E2 F899 B685 F815
9810  A5F8 22A6 D652 6422 3F18 371A 3014 0000
9820  0FFF F800 A7A8 5264 22F8 1FA4 4452 1404
9830  A9F7 AAAB 2424 2444 5214 0489 77FC 818A
9840  8BE9 1828 8873 2A9A 3A43 1988 F332 6078
9850  B7E2 F8EE D599 D589 D588 D597 F3D5 E97A
9860  1819 2B98 3A4B E217 873A 25D5 A0F8 8080
9870  E0D0
```

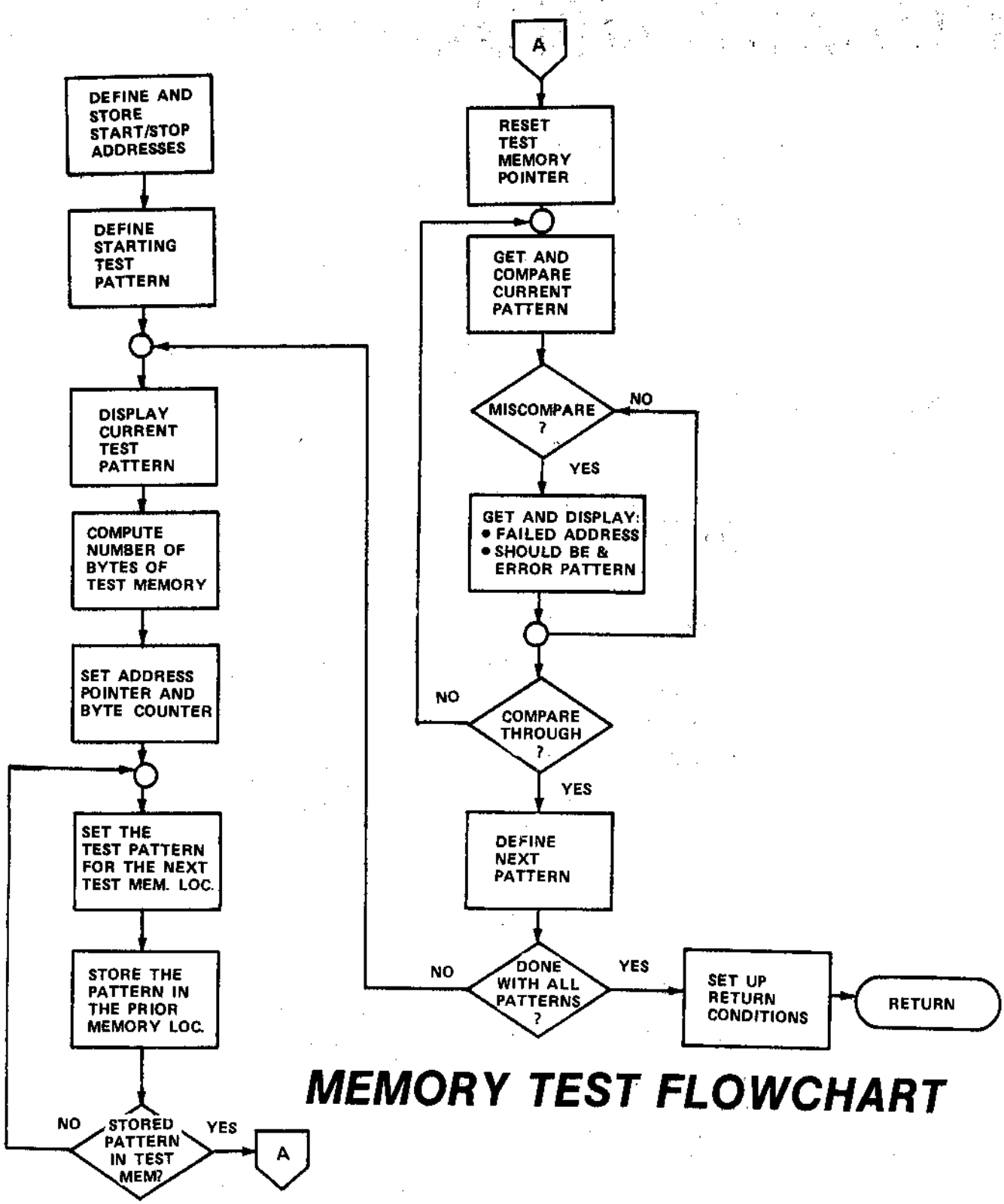
SHADES OF GRAY

By Eugene Jackson

Exciting visual effects will be yours when you load and run **SHADES OF GRAY.** The program displays 15 pages of RAM memory in slow to rapid succession (you choose the rate). Thus, if you load a box outline on page F of memory and fill in the box on pages E and D, an interesting effect will result. By experimenting with material entered in the COSMAC pages via the Super Monitor or other means, you can create blinking lights and 3-D effects. By putting the outline of trains, stoplights, cars, etc. on sequential pages in slightly different positions and places—animation can be produced using this program.

The speed of the sequence of memory pages displayed by the 1861 is determined by keypad entries while the program is running. Thus if you enter 01 you will get 60 pages per second (each page displayed four times per second!). On the other hand, if you enter FF you will get each page displayed steadily for five seconds at a whack. You will find it fun to experiment with different speeds and objects on the display pages.

LOC.	CODE	COMMENTS
0000	90 B1 B2 B3 B5 B6	
06	F8 1D A6	Pointer for current page
09	F8 32 A3	PC
0C	F8 4F A2	STACK
0F	F8 15 A1	Video interrupt routine pointer
12	D3	
13	72 70	EXIT (VID. INT. Routines)
15	22 78 22 52	ENTRY
19	C4 C4 C4	
1C	F8 0F B0	Load current pages address
1F	F8 00 A0	
22	80 E2	
24	E2 20 A0 E2 20 A0 E2 20 A0	
2D	3C 22 25 30 13 (Note the DEC of R5)	
32	E2 61 22	Turn on VIDEO
35	90 B7	
37	F8 10 A7	Load page counter (counts down)
3A	6C 64 22 A5	Load timer value
3E	85 3A 3E	Timer loop
41	27 87 32 35	
45	56 30 3A	Store current page in Loc. 001D

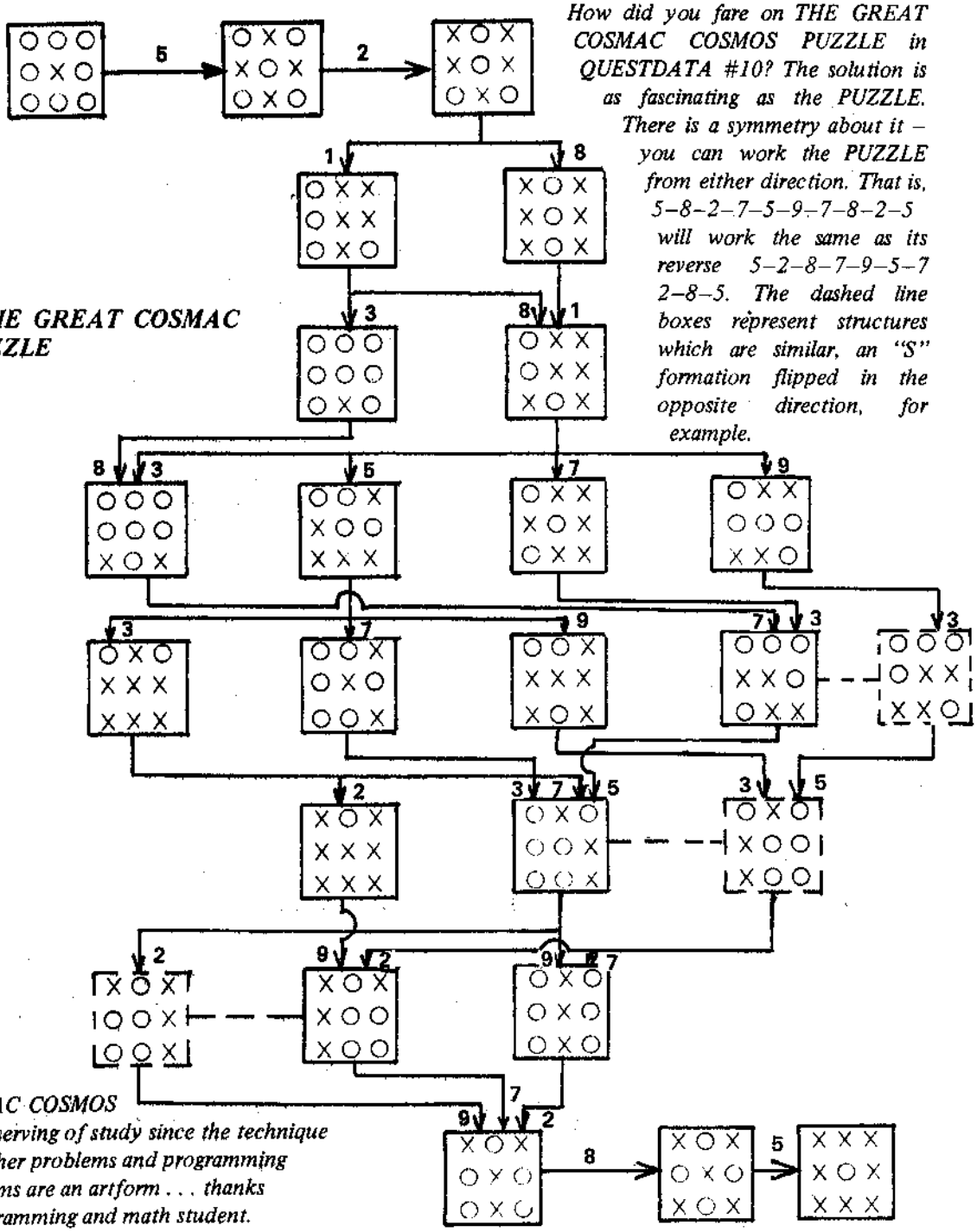


MEMORY TEST FLOWCHART

PUZZLE UNSCRAMBLED

Computer people are problem solvers. How did you fare on THE GREAT COSMAC COSMOS PUZZLE in QUESTDATA #10? The solution is as fascinating as the PUZZLE. There is a symmetry about it – you can work the PUZZLE from either direction. That is, 5–8–2–7–5–9–7–8–2–5 will work the same as its reverse 5–2–8–7–9–5–7 2–8–5. The dashed line boxes represent structures which are similar, an "S" formation flipped in the opposite direction, for example.

Solution to THE GREAT COSMAC COSMOS PUZZLE
By Reed Davis



The GREAT COSMAC COSMOS PUZZLE diagram is deserving of study since the technique can prove useful for other problems and programming situations. Clear diagrams are an artform . . . thanks go to Reed Davis, programming and math student.

COSMAC BATTLE OF NUMBERS

[Note: Floyd Oats felt the need to take a break from chess programming (who can blame him), and the result of his diversion is the following video puzzle which will run on 256 byte machines. He has used a modified version of the multipurpose Dzombak TVT. The lack of documentation describing how the program works is deliberate. If you knew how the brainteaser worked you would know the trick. Of course you could always trace through the code– making this a doubly challenging puzzle.]

By Floyd L. Oats

This game will run in practically any COSMAC computer, including the 256-byte basic ELF. It requires exactly one page of memory to run, with the last eighty bytes reserved for variables and I/O. The command code for the game is listed in Table 1, and execution should begin at address hex 00 with the P-Register equal to zero.

The computer will display forty spots at the lower portion of the screen, arranged as five rows of eight dots per row. The player takes turns with the computer in claiming these spots. Each, in his turn, will claim at least one but not more than four spots. Spots claimed by the computer will be replaced by short horizontal bars and spots claimed by the player will be replaced by long horizontal bars. He who claims the fortieth spot wins the game.

After starting the game, the first step is to decide who goes first. If the player wishes to go first, he enters his claim, hex 01 thru hex 04 (depending on how many he wishes to claim), on the keyboard or toggles and presses the INPUT button. The number of spots claimed will now be replaced by the long bars, and the computer will immediately display short bars to show how many it claims. To cause the computer to go first, enter hex FF and press the INPUT button. The computer will display the short bars and await your response. The machine will reject numbers outside the acceptable range.

The winner of the game is acknowledged on the hex display. If the computer wins, a hex CC will be displayed and the Q-LED will light. If the opponent wins, a hex 55 is displayed and the Q-LED is out.

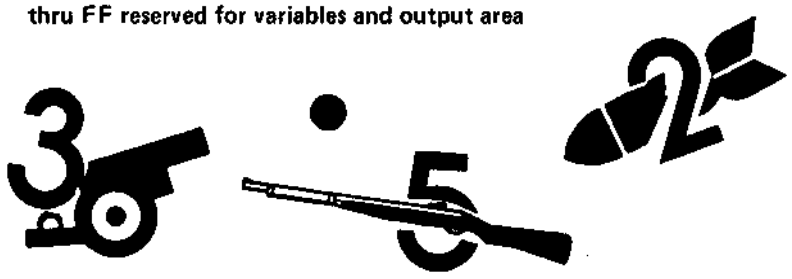
Even after you master the game, you can still have fun watching your friends try to beat it. People sometimes do funny things in attempts to defeat the computer.

I will leave you with two general rules which might help you to defeat the computer: (1) A SINGLE human error will, in ALL cases, result in a computer victory. (2) If the human player claims first, the computer has a WON GAME! Good luck!

COSMAC BATTLE OF NUMBERS

BY F.L. Oats

LOCATION	DATA															
0000	90	B1	B2	B3	F8	2E	A3	F8	B4	A2	F8	10	A1	D3	72	70
0010	22	78	22	52	C4	C4	C4	F8	00	A0	F8	00	B0	80	E2	E2
0020	20	A0	E2	20	A0	E2	20	A0	3C	1D	30	0E	00	00	E2	69
0030	F8	A7	AC	90	BF	BC	BA	F8	B0	AF	FA	08	32	42	F8	08
0040	30	44	F8	00	5F	1F	8F	3A	3A	A8	2F	F8	B8	AF	7A	EC
0050	1D	3F	50	37	53	6C	FB	FF	32	66	FC	01	33	50	FC	04
0060	3B	50	7A	0C	30	7B	31	4F	F8	A8	AA	EA	88	F5	33	73
0070	1A	30	6C	3A	7A	8D	FA	03	FC	01	7B	A9	31	82	F8	7E
0080	30	84	F8	1C	5F	1F	8F	FA	08	32	85	18	88	FB	28	32
0090	99	29	89	3A	7C	31	4F	30	68	31	A3	F8	55	EC	5C	64
00A0	2C	30	A1	F8	CC	30	9D	00	00	05	0A	0F	14	19	1E	23
00B0	thru FF reserved for variables and output area															



A FREE EDUCATION ON SLOT MACHINES

By Patrick E. Taylor

The idea and basic flowpath for SLOT MACHINE came from Radio Shack's ***Computer Programming in BASIC for Everyone*** (62-2015). Needless to say, the program underwent a significant rewrite.

SLOT MACHINE is a Tiny BASIC program in which the player bets $1 and then the Elf randomly selects a combination of three oranges, lemons or cherries. Three-of-a-kind wins $6, all other combinations lose. The Elf keeps track of the money and ends the game when you lose your last dollar. A sample run would look like this:

```
RUN
$1 SLOT MACHINE
PAYOFF IS $6 FOR
3 OF A KIND. ALL
OTHERS LOSE.
NUMBER OF DOLS.
TO START? 10
DO YOU WISH TO
PLAY (1 YES 0 NO)
?1
*OR**LM**CH*        TOO BAD YOU LOST


YOU NOW HAVE $9
DO YOU WISH TO
PLAY (1 YES 0 NO)
?1
*LM**LM**LM*        YOU WON $6

YOU NOW HAVE $15
DO YOU WISH TO
PLAY (1 YES 0 NO)
?0
SORRY ABOUT THAT
```

QUESTDATA
P.O. Box 4430
Santa Clara, CA 95054

Publisher . . . . . **Quest Electronics**
Editor . . . . . **Bill Haslacher**

Circulation . . . . . **Laurie Buzzell**

Listing of Program:

```
10   PRINT "$1 SLOT MACHINE"
20   PRINT "PAYOFF IS $6 FOR"
30   PRINT "3 OF A KIND. ALL"
40   PRINT "OTHERS LOSE."
50   PRINT "NUMBER OF DOLS."
60   PRINT "TO START";
70   INPUT M
80   LET X=RND(345)
90   PRINT "DO YOU WISH TO"
92   PRINT "PLAY (1 YES 0 NO)"
100  INPUT A
110  IF A=0 THEN GOTO 410
120  LET C=0
130  LET L=0
140  LET O=0
150  LET I=0
155 IF I=3 THEN GOTO 270
160  LET N=RND(3)+1
170  IF N=1 THEN GOTO 180
172  IF N=2 THEN GOTO 210
174  IF N=3 THEN GOTO 240
180  PRINT "*CH*";
190  LET C=C+1
200 GOTO 260
210  PRINT "*LM*";
220  LET L=L+1
230  GOTO 260
240  PRINT "*OR*";
250  LET O=O+1
260  LET I=I+1
265 GOTO 155
270  IF C=3 THEN GOTO 350
280  IF L=3 THEN GOTO 350
290  IF O=3 THEN GOTO 350
300  PRINT "        TOO BAD YOU LOST"
              (leave 8 spaces)
310  LET M=M-1
320  PRINT
330  IF M=0 THEN GOTO 400
340  GOTO 380
350  PRINT "        YOU WON $6"
              (leave 8 spaces)
360  LET M=M+6
370  PRINT
380  PRINT "YOU NOW HAVE $";M
390  GOTO 90
400  PRINT "NO MORE MONEY"
410  PRINT "SORRY ABOUT THAT"
420  END
```

QUESTDATA COSMAC CLUB

P.O. Box 4430, Santa Clara, CA 95054

ELECTRONIC DICE

By Ron Binning

This program uses a fast sequential count to simulate the roll of a pair of dice. Maximum length of time between addition of one to the value of the dice is about 3 ten-thousandths of a second, so it leaves to chance any roll.

Load program at location M(0000) or use ROM monitor. When using 32 byte ROM monitor, add starting location to M(17), M(21) and M(23).

To start your roll, push run (G) button and to stop the roll, push the wait (W) button.

LOC.	CODE	MNEM.	ACTION
0000	90	GHI 0	Initialization
01	B1	PHI 1	
02	F8	LDI	Pointer to wk.
03	FC		area
04	A1	PLO 1	R(1) points to 00FC
05	F8	LDI	Starting value of roll
06	11		is 11
07	51	STR 1	Mem. loc. FC=11
08	E1	SEX 1	X=1
09	64	OUT 4	Show loc. FC
0A	21	DEC 1	Restore address R(1)
0B	F0	LDX	D Reg. =Loc. ff
0C	FC	ADI	ADD 1 to
0D	01		D Reg.
0E	51	STR 1	Put result in loc. FC
0F	F0	LDX	D Reg. = Loc. FF
10	FE	SHL	Shift left
11	FE	SHL	to get rid of
12	FE	SHL	high order digit
13	FE	SHL	. . . (thus 4 SHL)
14	FB	XRI	Test
15	70		for 7
16	3A	BNZ	If D≠0
17	08		then GOTO 08
18	F0	LDX	D Reg. = Loc. FF
19	FC	ADI	ADD 11
1A	11		to adjust
1B	FF	SMI	Subtract 7
1C	07		to adjust units
1D	51	STR 1	Store in loc. FF
1E	FB	XRI	Test
1F	71		for 71
20	32	BZ	if D≠71
21	05		Then loc. 03
22	30	BR	If not 71
23	08		then loc. 08

WHAT's THE MYSTERY?

Why has Tiny BASIC source code been secret for so long? What hints to programmers lurk in the Tiny BASIC code? Find out by ordering your very own copy today. The complete source with Assembly listing is available form QUEST Electronics for $19.

By ordering your copy of the source code you will be able to modify Tiny BASIC code to suit your individual needs. You will also see how Tom Pittman uses a modified version of RCA's SCRT to write the code. QUEST Electronics has had many inquiries about the availability of the source code for Tiny BASIC by people who wish to modify or just read how the compact coding of Tiny was achieved. If you have been wondering "just what exactly is the Interpretive Language embedded in the Tiny BASIC code?"—you now have a chance to find out. There are many interesting tricks and routines you can adapt to other purposes. Now is your chance to find out exactly what the "mystery" source code is all about.

For a peek and poke at the inner workings of Tiny BASIC and the inventive mind of Tom Pittman, don't pass up this opportunity to buy the Tiny BASIC SOURCE CODE.

NUMBER GUESS

By Patrick E. Taylor

NUMBER GUESS is a Tiny BASIC program in which the Elf picks a random number and then players A and B try to guess which number the Elf has picked. Ten points are assigned to the player who's guess is the closest. The Elf keeps track of the score. A sample run would look like this:

```
RUN

NUMBER GUESS
PLAYER A VERSUS
PLAYER B. TEN
POINTS TO THE
WINNER. PICK A #
FROM 1 TO 100
PLAYER A? 25
PLAYER B? 78
ELF PICKED 83
PLAYER B WON!
A=0 B=10
LET'S TRY AGAIN
PLAYER A?

(and so on . . .)
```

Note: To end the game enter 0 for player A's guess.

Listing of Program:

```
1    PRINT "NUMBER GUESS"
2    PRINT "PLAYER A VERSUS"
3    PRINT "PLAYER B. TEN"
4    PRINT "POINTS TO THE"
5    PRINT "WINNER. PICK A #"
6    PRINT "FROM 1 TO 100"
15   LET F=0
16   LET G=0
20   PRINT "PLAYER A";
30   INPUT A
35   IF A=0 THEN GOTO 1000
40   PRINT "PLAYER B";
50   INPUT B
60   C=RND(100)
65   PRINT "ELF PICKED  ";
66   PRINT C
70   LET D=C-A
80   LET E=C-B
90   IF D*D>E*E THEN GOTO 500
100  PRINT "PLAYER A WON!"
101  LET F=F+10
110  GOTO 950
500  PRINT "PLAYER B WON!"
501  LET G=G+10
950  PRINT "A=";F
951  PRINT "B=";G
990  PRINT "LET'S TRY AGAIN"
991  GOTO 20
1000 END
```

COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC CLUB COSMAC

11 QUESTDATA
P.O. Box 4430
Santa Clara, CA 95054

ADDRESS CORRECTION REQUESTED

BULK RATE
U.S. Postage Paid
QUEST
Electronics

Permit No. 549
Santa Clara, CA