to start the timer. If the first try fails, the second will work since the timer will not have counted back to zero by the time the second store is executed. If the first try is successful then the second will also be successful since the timer will not have counted back to zero yet. Does this bug cause any problems in the AIM? I am not sure. I looked at the AIM program listing manual and found that the AIM uses this timer in the printer routines and the tape routines. The best guess I could make is that it may cause an occasional 'PRINTER DOWN' when the printer is really up, or maybe a lost bit on tape. I would like to know if anyone has experienced any problems like these.

AIM 65 provides you with a flexible I/O system. The user I/O function gives an expandability not usually found in an SBC. If you follow the guidelines given here you should be able to implement any device via the AIM user I/O function.

Dungeons And Dragons Dice Simulator For The KIM-I

Myron A. Calhoun
Manhattan, KS

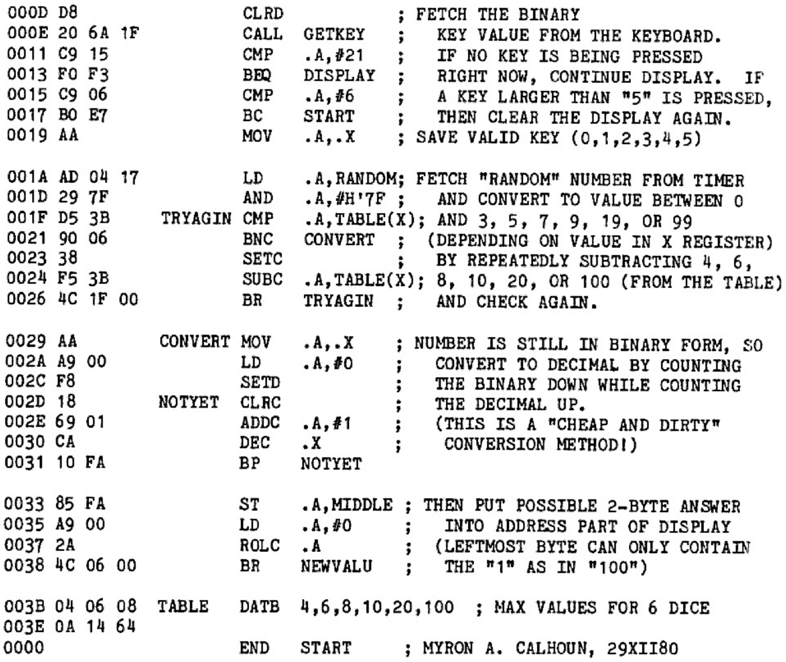
Last Christmas my older son received a "Dungeons and Dragons" game, but when the package was opened there were no dice included. (There were small numbered squares of cardboard for shaking and drawing out of a cup, but this seemed to slow the game considerably.) Even worse, the local hobby shop was completely sold out, so a state of near emergency existed.

(Trumpets!) Enter a KIM-1 to the rescue. The enclosed little program was quickly derived, my son was taught how to load it into the KIM-1, and the crisis was over. (Even though he has since located dice in another store, their relatively high cost and his small allowance have caused him to continue using this program!)

Summary Of Operation

Pressing the KIM-1's "0", "1", "2", "3", "4", or "5" key simulates rolling a 4-, 6-, 8-, 10-, 20-, or 100-sided die. The result is displayed as a random number in the range 1-4, 1-6, 1-8, 1-10, 1-20, or 1-100, respectively. Pressing any other key clears the display to zeroes. Holding any one of the operational keys down displays successive random numbers but too fast to read. "Random" numbers are derived from the free-running built-in timer in the KIM-1.

```
ASM6502:  6502 CROSS-ASSEMBLER USING PROPOSED I.E.E.E. STANDARD (DRAFT 11)
               ; DUNGEONS AND DRAGONS DICE SIMULATOR FOR THE KIM-1

               ; PRESSING THE KIM-1'S "0", "1", "2", "3", "4", OR "5" KEY
               ; SIMULATES ROLLING A 4-, 6-, 8-, 10-, 20-, OR 100-SIDED
               ; DIE.  THE RESULT IS DISPLAYED AS A RANDOM NUMBER IN THE
               ; RANGE 1-4, 1-6, 1-8, 1-10, 1-20, OR 1-100, RESPECTIVELY.
               ; HOLDING ONE OF THE ABOVE-NAMED KEYS DOWN WILL DISPLAY
               ; SUCCESSIVE RANDOM NUMBERS BUT TOO FAST TO READ.  PRESSING
               ; ANY OTHER KEY WILL CLEAR THE DISPLAY TO ZEROES.  "RANDOM"
               ; NUMBERS ARE DERIVED FROM THE FREE-RUNNING BUILT-IN TIMER
               ; IN THE KIM-1.

1704           RANDOM  EQU  H'1704     ; DEFINE MISCELLANEOUS ADDRESSES
00FB           LEFT    EQU  H'FB       ;   OF THIS-N-THAT IN THE KIM-1
00FA           MIDDLE  EQU  LEFT-1     ;   "OPERATING SYSTEM" RESERVED
00F9           RIGHT   EQU  MIDDLE-1   ;   MEMORY AREA
1F1F           SCANDS  EQU  H'1F1F
1F6A           GETKEY  EQU  H'1F6A

0000 A9 00     START   LD   .A,#0      ; CLEAR THE INITIAL DISPLAY
0002 85 F9             ST   .A,RIGHT   ;   TO ALL ZEROES
0004 85 FA             ST   .A,MIDDLE

0006 85 FB     NEWVALU ST   .A,LEFT    ; SET NEW VALUE (FOUND BELOW)

0008 20 1F 1F  DISPLAY CALL  SCANDS    ; "PUMP" THE DISPLAY AND ALSO
000B F0 FB             BZ    DISPLAY   ;   SEE IF ANY KEYS ARE PRESSED
```



```
000D D8                 CLRD                ; FETCH THE BINARY
000E 20 6A 1F           CALL  GETKEY    ;   KEY VALUE FROM THE KEYBOARD.
0011 C9 15              CMP   .A,#21    ;   IF NO KEY IS BEING PRESSED
0013 F0 F3              BEQ   DISPLAY   ;   RIGHT NOW, CONTINUE DISPLAY.  IF
0015 C9 06              CMP   .A,#6     ;   A KEY LARGER THAN "5" IS PRESSED,
0017 B0 E7              BC    START     ;   THEN CLEAR THE DISPLAY AGAIN.
0019 AA                 MOV   .A,.X     ; SAVE VALID KEY (0,1,2,3,4,5)

001A AD 04 17           LD    .A,RANDOM; FETCH "RANDOM" NUMBER FROM TIMER
001D 29 7F              AND   .A,#H'7F ;   AND CONVERT TO VALUE BETWEEN 0
001F D5 3B     TRYAGIN  CMP   .A,TABLE(X); AND 3, 5, 7, 9, 19, OR 99
0021 90 06              BNC   CONVERT  ;  (DEPENDING ON VALUE IN X REGISTER)
0023 38                 SETC            ;   BY REPEATEDLY SUBTRACTING 4, 6,
0024 F5 3B              SUBC  .A,TABLE(X); 8, 10, 20, OR 100 (FROM THE TABLE)
0026 4C 1F 00           BR    TRYAGIN  ;   AND CHECK AGAIN.

0029 AA        CONVERT  MOV   .A,.X    ; NUMBER IS STILL IN BINARY FORM, SO
002A A9 00              LD    .A,#0    ;   CONVERT TO DECIMAL BY COUNTING
002C F8                 SETD           ;   THE BINARY DOWN WHILE COUNTING
002D 18        NOTYET   CLRC           ;   THE DECIMAL UP.
002E 69 01              ADDC  .A,#1    ;   (THIS IS A "CHEAP AND DIRTY"
0030 CA                 DEC   .X       ;    CONVERSION METHOD!)
0031 10 FA              BP    NOTYET

0033 85 FA              ST    .A,MIDDLE ; THEN PUT POSSIBLE 2-BYTE ANSWER
0035 A9 00              LD    .A,#0     ;   INTO ADDRESS PART OF DISPLAY
0037 2A                 ROLC  .A        ;  (LEFTMOST BYTE CAN ONLY CONTAIN
0038 4C 06 00           BR    NEWVALU   ;   THE "1" AS IN "100")

003B 04 06 08  TABLE    DATB  4,6,8,10,20,100  ; MAX VALUES FOR 6 DICE
003E 0A 14 64
0000                    END   START     ; MYRON A. CALHOUN, 29XII80
```

The program is written using the proposed IEEE (Institute of Electrical and Electronic Engineers) Microprocessor Assembly Language Standard (Draft 11) as it applies to the 6502 microprocessor. Although it differs slightly from the assembly language seen in other **COMPUTE!** articles, it should be easily understandable. According to Wayne P. Fischer, Chairman of the IEEE Computer Society's Microprocessor Assembly Language Standard Subcommittee, "The impetus for the development of this standard was the helter skelter proliferation of microprocessor mnemonic codes, the inconsistent and conflicting use of operands, the varying definition of address modes, and other annoying anomalies of the various assembly languages. The standard will transform this mishmash of languages into one that is consistent, easily understood, and easily used"(1).

The program itself is rather simple and the comments should explain it sufficiently. About the only "trick" is the method used to convert a binary number in the accumulator into a BCD number in the display. Beginning at the label CONVERT (at address H'0029), the program performs a "brute force" conversion by counting the binary value downward (after moving it to the index register) while simultaneously counting the BCD value upward in the accumulator *in decimal mode*. The value 100 (decimal) causes the CARRY bit to be set, and care must be taken to move the "1" to the display.

The TABLE values (at location H'003B) may be changed if other maximum die values are desired. The maximum length of the table is the immediate operand of the instruction at location H'0015.

The program is short enough that loading before a game takes but a few minutes. It has even gotten my boy a little interested in computers!

(1) Fisher, Wayne P., "Microprocessor Assembly Language Draft Standard", IEEE Computer Magazine, December 1979, pp. 96-109.

©