SPECIAL FOCUS ON computers...

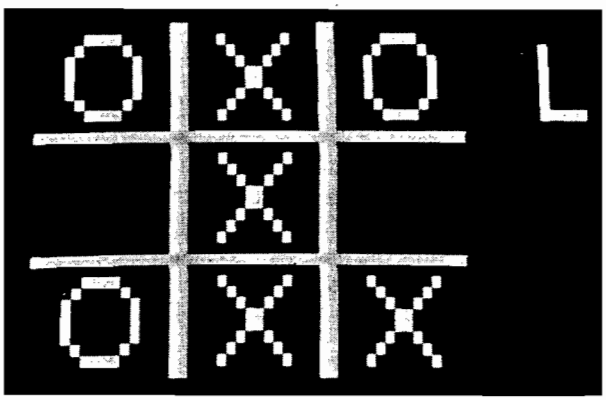
Here's an example of how the Aux Box can be used in conjunction with a cassette tape system. Assume that the assigned location of a program on tape is at 150 on the index counter. Rewind the tape by setting *S2* to RECORDER and *S3* to TAPE SPEED. Latch the recorder's REWIND lever and press *S4* (FAST switch) to rewind a short tape. With long tapes, use *S3* in the MIC position for rewinding. Reset the index counter and then operate the recorder's STOP lever.

To position the tape, press the recorder's FAST-F lever and latch it. Press *S4* (FAST) and observe the counter's 10's and 100's digits. When 14X appears on the counter, release *S4*. The tape will probably coast to about 146. At this point, you must select between immediate use of the monitor (PLAY lever engaged) or defer use of the monitor (PLAY lever disengaged) and move the tape to 150 on the index counter, using *S5* (MEDIUM) or *S6* (SLOW).

The tape can be run back and forth to access other programs without rewinding or resetting the index counter. Going backward (rewinding) from 175 to 20, use the REWIND lever on the recorder and *S4* (FAST switch) on the Aux Box, letting go of *S4* when 03X appears in the index counter. Use SLOW reverse to jog the tape to its final position. On the MEDIUM speed, the counter's 1's decade is at least readable and tape coasting is reduced by one-half. This can be used to great advantage.

Summing Up. If you store many programs on one cassette tape, the Aux Box lets you see just how easy it can be to retrieve programs quickly and precisely. Without having to pull cables, you can transfer control between the computer and tape deck at the flip of a switch and monitor the tape at all times. And you can transport tape at any of three speeds in either direction. In short, the Aux Box takes much of the hassle out of loading programs. ◇

A TIC-TAC-TOE GAME for your Elf Computer



Use a simple light pen as an input selector and the programs given here.

BY EDWARD M. McCORMICK

AN ELF computer that contains 1K of RAM can be programmed to play Tic-Tac-Toe if it is equipped with a video (1861) display and a light pen, the latter to be described here. The computer, using an O, plays against the human (X), and either the computer or the human can go first.

Unless forced to make some other play, the computer will randomly select any open position as its response. This results in a wide variety of games, and although the computer can be beaten, it will also win more than one might expect.

Playing the Game. The playing sequence is straightforward. The IN switch is operated to clear the screen. If any toggle switch is on, the computer will indicate its O after a pause of two seconds. If the input toggle switches are all off, the computer will wait for you to make the first move.

Whenever you are to play, a P will be displayed at the upper-right corner of the screen. You place the light pen in front of the position you wish to play, then depress the IN switch. An X will appear at the selected position. The computer will then indicate its response and you play again. This cycle continues until the game ends.

A D for draw, W for win, or L for lose will be shown at the upper-right corner at the end of a game. To clear the screen and set up for another game, simply operate the IN switch.

The Light Pen. The light pen consists of a cadmium-sulphide cell (Radio Shack 276-116) connected between EF-3 and ground in the Elf. Make sure that a 47,000-ohm resistor is connected between EF3 and +5 volts.

The main program, subroutines, and data sets are given in Tables I through V. Program execution starts at 0400. Table VI indicates the initial contents of page 6, the display area. Note that this forms the familiar Tic-Tac-Toe grid.

The program requires about 525 bytes, including main program, subroutines, and data storage. In addition, 256 bytes are used for the display area. The remainder of the RAM space can be used for embellishments to the program.

Initially, the program sets up the registers used. After the IN switch has been operated, the nine playing positions and the status position are cleared and N (the number of plays) is set to zero. If the input toggle switches are not set to zero, the program goes to the "any place" position (Table I). The computer randomly selects one of the unplayed positions and places an O in that location. The program then adds a 1 to N and checks to see if all nine positions have been played. If not, it allows the opponent to play an X. Note that when the input toggle switches are at zero, the program goes directly to X.

TABLE I—MAIN PROGRAM

Loc.	Program	Comments
04 00	C0 05 20	Initialize Registers
04 03	E9 69	Start video display
04 05	3F 05 37 07	Operate IN Switch
04 09	F8 E0 A3	Clear All Positions
04 0C	E3 F8 14 A5	including 9 playing
04 10	F0 32 18	and the status
04 13	A6 D4 13	position
04 16	30 0C	
04 18	F8 00 A8	N = O
04 1B	E9 6C 32 4D	Toggle .Sw On?
04 1F	8B AC 8C	Any Place
04 22	FF 09 32 2A	Determine random
04 26	33 22 FC 09	position to play,
04 2A	FC E0 A3	if occupied go
04 2D	E3 F0 A6 E6	back for another
04 31	F0 3A 1F	random position
04 34	F8 80 59 DD	'O'
04 38	F8 24 A5 D4	Write 'O'
04 3C	88 FC 01 A8	N = N + 1
04 40	FF 09 3A 4D	N = 9?
04 44	F8 44 A5	'D'
04 47	F8 0F A6 D4	If N=9, write 'D',
04 4B	30 05	return to restart
04 4D	F8 4C A5	Operate IN Switch
04 50	F8 0F A6 D4	Write 'P' then
04 54	3F 54 37 56	wait for IN switch
04 58	F8 01 59 DD	'X'
04 5C	F8 00 A3	If area bright
04 5F	36 6B	then skip to turn on
04 61	83 FC 01 A3	'?' and retry after
04 65	FF 20 3A 5F	short wait
04 69	30 78	
04 6B	F8 2C A5	Put '?' in status
04 6E	F8 0F A6 D4	
04 72	F8 20 59 DD	
04 76	30 5C	
04 78	F8 14 A5	If area dark, start
04 7B	F8 0F A6 D4	scanning
04 7F	F8 00 A7	Get playing position
04 82	FC E0 A3 E3	from table
04 86	F0 A6 BC	If it is occupied
04 89	E6 F0 32 A1	get next position
04 8D	87 FF 08	If all unoccupied
04 90	32 98	positions scanned but
04 92	87 FC 01 A7	not recognized, turn
04 96	30 82	on
04 98	F8 2C A5	'?' as status but
04 9B	F8 0F A6 D4	continue
04 9F	30 7F	scan
04 A1	F8 1C A5 D4	Write 'X' in
04 A5	F8 01 59 DD	unoccupied position,
04 A9	F8 00 A3	make tries to see
04 AC	36 BE	if it is pen
04 AE	83 FC 01 A3	light position
04 B2	FF 20 3A AC	
04 B6	9C A6	If not recognized
04 B8	F8 14 A5 D4	erase 'X' and go to
04 BC	30 8D	next position.
04 BE	F8 14 A5	When 'X' recognized
04 C1	F8 0F A6 D4	clear status.
04 C5	F8 1B A5 DA	Lost?
04 C9	86 32 D5	If so, write
04 CC	F8 3C	'L'
04 CE	A5 F8 0F A6	and return
04 D2	D4 30 05	to restart
04 D5	88 FC 01 A8	N = N + 1
04 D9	FF 09 32 44	N = 9?
04 DD	F8 06 A5 DA	Win?
04 E1	86 32 F0	If so, write
04 E4	F8 80 59 DD	'O'
04 E8	F8 24 A5 D4	to win then write
04 EC	F8 34 30 CE	'W'
04 F0	F8 12 A5 DA	Def?
04 F4	86 32 1F	If so, play that 'O',
04 F7	30 34	else go to Any Place

TABLE II—CHARACTER PRINT SUBROUTINE WITH DATA FOR CHARACTERS USED.

Loc.	Program	Comments
07 00	DF E5 F8 08 AC F0 56	Write reg 5 8 byte
07 07	8C FF 01 AC 32 00	* character into reg 6
07 0D	15 86 FC 08 A6 30 05	* position on screen
07 14	00 00 00 00 00 00 00 00	Blank
07 1C	81 42 24 18 18 24 42 81	X
07 24	3C 42 81 81 81 81 42 3C	O
07 2C	00 0E 11 02 04 04 00 04	?
07 34	00 11 11 11 15 15 1B 11	W
07 3C	00 10 10 10 10 10 10 1F	L
07 44	00 1E 11 11 11 11 11 1E	D
07 4C	00 1E 11 11 1E 10 10 10	P

TABLE III—DELAY SUBROUTINE

Loc.	Program
05 F0	DF E9 8B F4 AC 59
05 F6	8B F7 3A F6 30 F0

Immediately after the opponent has played, the computer checks to see if it has lost. If it has lost, the computer displays an L and waits for the IN switch to be operated to start the next game. If the computer has not lost, it adds 1 to N to see if all nine positions have been played and, if so, displays a D and awaits a new game.

If the computer has not lost and the game is not a draw, the computer checks to determine if it can win. All eight sets of positions are checked to find one that has two O's and the third position blank. If it finds such a combination, it will display an O in that third position, display a W and return to restart.

If the computer has neither lost nor won, the program then takes defensive action. The eight sets of positions are examined to see if the opponent has two X's and the third position blank. If it finds such a situation, the computer inserts an O at that position, and continues. If not compelled to make a defensive play, the

TABLE IV—LOSE-WIN-DEFENSIVE (L-W-D) SUBROUTINE

Loc.	Program	Comments
05 60	DF F8 C0 A7	Set addr init position
05 64	F8 00 A3 AC	Clear counters
05 68	E7 F0 AE EE F0 32 7F	Examine location
05 6F	FF 81 32 79	* addressed,
05 73	8C FC 03 AC 30 81	* if 'O' add 3,
05 79	8C FC 09 AC 30 81	* if 'X' add 9, else
05 7F	8E A6	* store blank position
05 81	17 83 FC 01 A3	Repeat for all 3 positions
05 86	FF 03 3A 68	* of combination
05 8A	85 59 E9 8C F7 32 9B	If L-W-D match not found,
05 91	87 FF D8 3A 64	* go to next combination
05 96	F8 00 A6 30 60	Exit if all 8 combs fail
05 9B	86 3A 60	Set register 6 and exit
05 9E	F8 01 A6 30 60	* if L-W-D match found
05 C0	09 0B 0D 09 63 BD	The eight combinations
05 C6	09 61 B9 0B 63 BB	* of three positions
05 CC	0D 63 B9 0D 65 BD	* (rows, columns
05 D2	61 63 65 B9 BB BD	* and diagonals)
05 E0	63 09 0D B9 BD	The 9 playing and the
05 E5	0B 61 65 BB 0F 00	* status positions

TABLE V— INTERRUPT ROUTINE FOR THE 1861 CHIP

Loc.	Program
05 00	72 70 22 78 22 52
05 06	C4 C4 C4
05 09	F8 06 B0 F8 00 A0
05 0F	80 E2
05 11	E2 20 A0 E2 20 A0
05 17	E2 20 A0
05 1A	3C 0F 1B 30 00

TABLE VII—REGISTER USE IN TTT PROGRAM

Register	Use
0	Interrupt DMA pointer
1	Interrupt routine
2	Stack pointer
3	Page 5 pointer
4	Character print subr
5	'From' char pointer
6	'To' char pointer
7	Page 5 pointer
8	N, number of plays
9	Temp storage pointer
A	L-W-D subr
B	Refresh count
C	Temp storage
D	Delay subr
E	Win, Lose, Draw
F	Main program

TABLE VI—DISPLAY AREA IN PAGE 6 WITH TTT GRID

Loc.	Data
06 00	00 00 18 00 18 00 00 00
06 08	00 00 18 00 18 00 00 00
06 10	00 00 18 00 18 00 00 00
06 18	00 00 18 00 18 00 00 00
06 20	00 00 18 00 18 00 00 00
06 28	00 00 18 00 18 00 00 00
06 30	00 00 18 00 18 00 00 00
06 38	00 00 18 00 18 00 00 00
06 40	00 00 18 00 18 00 00 00
06 48	00 00 18 00 18 00 00 00
06 50	07 FF FF FF FF FF E0 00
06 58	00 00 18 00 18 00 00 00
06 60	00 00 18 00 18 00 00 00
06 68	00 00 18 00 18 00 00 00
06 70	00 00 18 00 18 00 00 00
06 78	00 00 18 00 18 00 00 00
06 80	00 00 18 00 18 00 00 00
06 88	00 00 18 00 18 00 00 00
06 90	00 00 18 00 18 00 00 00
06 98	00 00 18 00 18 00 00 00
06 A0	00 00 18 00 18 00 00 00
06 A8	07 FF FF FF FF FF E0 00
06 B0	00 00 18 00 18 00 00 00
06 B8	00 00 18 00 18 00 00 00
06 C0	00 00 18 00 18 00 00 00
06 C8	00 00 18 00 18 00 00 00
06 D0	00 00 18 00 18 00 00 00
06 D8	00 00 18 00 18 00 00 00
06 E0	00 00 18 00 18 00 00 00
06 E8	00 00 18 00 18 00 00 00
06 F0	00 00 18 00 18 00 00 00
06 F8	00 00 18 00 18 00 00 00

TABLE VIII—INITIALIZATION OF THE REGISTERS

Loc.	Program
05 20	F8 05 BF F8 27 AF DF
05 27	F8 05 B1 B2 B3
05 2C	B7 B9 BA BD F8 06
05 32	B6 BE F8 07 B4 B5
05 38	F8 02 A1 F8 BF A2
05 3E	F8 E0 A3 F8 01 A4
05 44	F8 FF A9 F8 61 AA
05 4A	F8 F1 AD C0 04 03

program returns to "any place" and randomly selects its next response.

If the light pen is placed in front of an already lit position when it is the opponent's turn to play, a ? will be displayed. As soon as the pen is moved to a dark position, the program will start scanning all unplayed positions to display the X. If the light pen is away from all positions, a ? will be displayed. The ? is erased when a valid X is written.

Program. The program consists of the Main Program shown in Table I and four subroutines. These are Character Print in Table II, Delay in Table III, L-W-D (Lose-Win-Defensive) in Table IV, and Video Chip Interrupt in Table V.

The character-print subroutine can print a blank or any of the seven characters X, O, ?, W, L, D, or P in any of the 10 positions on the screen. The delay subroutine is used for various delays including the 2-second delay before displaying an O. The L-W-D subroutine is used for the three tests to determine if the computer has lost, can win, or must play defensively. For each row, column, or diagonal, the computer examines each of the three positions. If it is an X, nine is added to a register; if a O, three is added; and if blank, nothing is added. Thus, a total of 27 indicates a loss, a total of 6 a potential win, and a total of 18 indicates that defensive action is required. The video interrupt instruction is conventional.

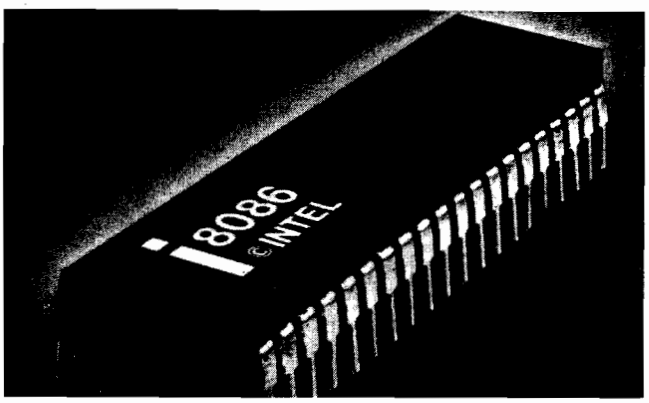
The random position for "any place" is determined by the B register, which is used to count the number of refreshes of the TV screen. Whenever "any place" is entered, the program takes the modulo-9 value of this count for the position to be played. If that position is already occupied, another number is generated and the process is repeated until an empty position is found.

The use of the various registers is shown in Table VII.

The first seven bytes of the register initialization section (Table VIII) makes the switch if the program register upon entry is not F. If it is F, this part of the initialization must be skipped. ◇

16-Bit VS 8-Bit Microprocessors

BY GORDON LETWIN&HAMPTON MILLER
Heath Co.



8086 CPU is part of Intel's new 16-bit family.

IT IS OUR purpose here to provide a brief comparison of the recently introduced 16-bit microprocessors with their predecessors, the 8-bit CPU's. While we cannot do full justice to a complete comparison of all aspects of the two types, some broad generalizations can be made that will give a good idea of what to expect from the new 16-bit machines.

The Differences. Microprocessors deal with elementary units of digital information called "bits." An 8-bit processor's registers and data paths are all eight bits wide. Therefore, it operates on data and instructions eight bits (one byte) at a time. This does not mean that an 8-bit processor cannot deal with data larger than eight bits wide. The data can be broken up into 8-bit bytes and processed one byte at a time.

Many processor instructions, such as branches and memory references, are 16 and 24 bits long and are known as 2- and 3-byte instructions. These processors may also contain a few instructions that deal directly with 16-bit values. However, these multibyte instructions and 16-bit operations are handled eight bits at a time and, therefore, take more time to execute than do single-byte instructions dealing with 8-bit values. Typical applications are process control and communication, tasks that require minimal computation.

In contrast to the 8-bit processor, the registers and data paths of a 16-bit processor are all 16-bits wide. Consequently, the 16-bit processor operates on data and instructions 16 bits at a time. Since the number of operations per second is roughly the same for both 8- and 16-bit processors, two times as much work gets done per instruction with a 16-bitter. Hence, a 16-bit processor can be much faster than an 8-bitter.

The 16-bit processor is more than a double-sized 8-bit device. Since the 16-bitter fetches instructions from memory 16 bits at a time, its instruction set usually includes 16-, 32-, and 48-bit instructions. The longer instructions allow a 16-bit machine to implement sophisticated instructions that tend to be more general-purpose to take better advantage of a given architecture than does the 8-bit device. For example, to interpret the BASIC statement A=B(4) with an 8-bit 8080A, the BASIC interpreter might execute the code as follows:

```
LXI   H,B
LXI   D,4
DAD   D     ;(HL)=address of value
MOV   E,M
INX   H
MOV   D,M   ;(DE)=value
XCHG
SHLD  A     ;store in A
```

On the other hand, a 16-bit computer such as an LSI-11 could use the following code:

```
MOV   #4,R1     ; (R1)=subscript
MOV   B(R1),A   ;look up and store
```

Note how the 16-bit machine avoids the clumsy and time-consuming memory-access gyrations required by the 8-bit device.

Although 8-bit devices deal with smaller values than do 16-bit devices, this does not mean that the former cannot perform all the computations that a 16-bitter can perform. It just requires more time because calculations must be processed in smaller pieces that must be "fetched" sequentially from memory. To perform the operation A=B-C with the 8080A requires the following code:

```
LHLD  C
XCHG
LHLD  B
MOV   A,L
SUB   E     ; subtract low 8 bits
MOV   L,A
MOV   A,H
SBB   D     ; subtract high 8 bits
MOV   H,A
SHLD  A     ; store result
```