A MATTER OF TIMING

Sometimes the timing *of an assembly language program is critical. Learn how to count clock cycles and apply your knowledge to producing music.*

The main advantage of assembly language over BASIC or Pascal is *speed*. In fact, the main advantage of computers over hand calculators and typewriters is speed.

But up until now, speed has been conspicuously absent from our discussion of the 6502/65C02 opcodes. Let's rectify that situation right now.

IT'S ABOUT TIME

Boot your assembler/editor, enter the short assembly language program in **Listing 1**, assemble it, and save the source and object codes to disk under the base name CLOCK.TEST. If you don't have an assembler, you can enter the Monitor with CALL −151 and key in the hex code. Save the program with the command:

BSAVE CLOCK.TEST,A$300,L$2C

For help with entering *Nibble* listings, see the beginning of the Listings Section at the back of this issue.

Now, leave your assembler and get into BASIC. Get out your stopwatch, type **BLOAD CLOCK.TEST**, and run the program (**CALL 768**). After an initial short pause, you will hear a beep, at which point you should start your watch. When you hear a second beep, stop your watch and record the time (preferably to 0.01 of a second). Repeat this process three or four times, and take an average of the results.

The actual time from the beginning of one beep to the beginning of the next is 5.10 seconds.

Clock Rate

You could determine this exact value by repeating the stopwatch test a hundred times, but there is an easier, more accurate way: calculate the execution time of each CLOCK .TEST instruction.

To determine the time required for each 6502/65C02 instruction in any machine language program, you need to know two things:

1. The overall clock rate of the Apple's microprocessor (in cycles per second)
2. The number of clock cycles required for each instruction.

The clock rate is the frequency at which the microprocessor performs each of its tasks. For the Apple II Plus, IIe and IIc, *the clock rate is exactly 1.02048432 MHz* (MHz means megahertz, or million cycles per second), or approximately 1 MHz.

Note: By comparison, the clock rates of other microcomputers (and their microprocessors) are: Apple IIGS (65816 processor)-2.5 MHz; Macintosh (68000)-8 MHz; IBM PC (8088)-4.77 MHz; IBM AT (80286)-8 MHz.

Notice that the Apple clock rate is determined by hardware circuitry on the motherboard, and has nothing to do with a clock/calendar card, which you can plug into an Apple IIe slot for date/time stamping of ProDOS files.

The *Apple II Technical Reference Manual* lists the clock rate as 1.023 MHz (or more accurately, 1.022727143 MHz), which is the frequency of the *normal* clock cycle. But this value does not take into account the so-called "long cycle," which occurs once every 65 clock cycles (required for proper video scanning). The long cycle brings the average clock rate down to 1.02048432 MHz. It also causes clock-pulse jitter (uneven timing), which you can ignore except in applications requiring extremely accurate timing. For a full discussion of Apple timing, see *Understanding the Apple II* by Jim Sather, Quality Software, 21601 Marilla St., Chatsworth, CA 91311.

TABLE 1: The 6502/65C02 Instruction Set

The Processor Flags column indicates the status flags affected by the particular mnemonic and applies to all addressing modes of that mnemonic. The Notes column refers to the numbered notes at the end of the table.

Instr. Mnemonic	Addressing Mode	Hex Opcode	Clock Cycles	Number Bytes	Processor Flags	Notes
ADC	Immediate	69	2	2	NV....ZC	(1)
	Absolute	6D	4	3		(1)
	Zero page	65	3	2		(1)
	Indexed indirect	61	6	2		(1,2)
	Indirect indexed	71	5	2		(1,2)
	Zero pg. index X	75	4	2		(1,2)
	Absolute index X	7D	4	3		(1,2)
	Absolute index Y	79	4	3		(1,2)
	Zero pg. indirect	72	5	2		(1,2,4)
AND	Immediate	29	2	2	N.....Z.	
	Absolute	2D	4	3		
	Zero page	25	3	2		
	Indexed indirect	21	6	2		(2)
	Indirect indexed	31	5	2		(2)
	Zero pg. index X	35	4	2		(2)
	Absolute index X	3D	4	3		(2)
	Absolute index Y	39	4	3		(2)
	Zero pg. indirect	32	5	2		(2,4)
ASL	Absolute	0E	6	3	N.....ZC	
	Zero page	06	5	2		
	Accumulator	0A	2	1		
	Zero pg. index X	16	6	2		(2)
	Absolute index X	1E	6	3		(2)
BCC	Relative	90	2	2	....	(3)
BCS	Relative	B0	2	2	....	(3)
BEQ	Relative	F0	2	2	....	(3)
BIT	Immediate	89	2	2	NV....Z.	(4)
	Absolute	2C	4	3		
	Zero page	24	3	2		
	Zero pg. index X	34	4	2		(4)
	Absolute index X	3C	4	3		(4)
BMI	Relative	30	2	2	....	(3)
BNE	Relative	D0	2	2	....	(3)
BPL	Relative	10	2	2	....	(3)
BRA	Relative	80	2	2	....	(3,4)
BRK	Implied	00	7	1	...B.I..	
BVC	Relative	50	2	2	....	(3)
BVS	Relative	70	2	2	....	(3)
CLC	Implied	18	2	1	C	
CLD	Implied	D8	2	1	....D...	
CLI	Implied	58	2	1	.....I..	
CLV	Implied	B8	2	1	.V......	
CMP	Immediate	C9	2	2	N.....ZC	
	Absolute	CD	4	3		
	Zero page	C5	3	2		
	Indexed indirect	C1	6	2		(2)
	Indirect indexed	D1	5	2		(2)
	Zero pg. index X	D5	4	2		(2)
	Absolute index X	DD	4	3		(2)
	Absolute index Y	D9	4	3		(2)
	Zero pg. indirect	D2	5	2		(2,4)
CPX	Immediate	E0	2	2	N.....ZC	
	Absolute	EC	4	3		
	Zero page	E4	3	2		
CPY	Immediate	C0	2	2	N.....ZC	
	Absolute	CC	4	3		
	Zero page	C4	3	2		
DEC	Absolute	CE	6	3	N.....Z.	
	Zero page	C6	5	2		
	Accumulator	3A	2	1		(4)
	Zero pg. index X	D6	6	2		(2)
	Absolute index X	DE	6	3		(2)
DEX	Implied	CA	2	1	N.....Z.	
DEY	Implied	88	2	1	N.....Z.	
EOR	Immediate	49	2	2	N.....Z.	
	Absolute	4D	4	3		
	Zero page	45	3	2		
	Indexed indirect	41	6	2		
	Indirect indexed	51	5	2		
	Zero pg. index X	55	4	2		
	Absolute index X	5D	4	3		
	Absolute index Y	59	4	3		
	Zero pg. indirect	52	5	2		(4)
INC	Absolute	EE	6	3	N.....Z.	
	Zero page	E6	5	2		
	Accumulator	1A	2	1		(4)
	Zero pg. index X	F6	6	2		(2)
	Absolute index X	FE	6	3		(2)
INX	Implied	E8	2	1	N.....Z.	
INY	Implied	C8	2	1	N.....Z.	
JMP	Absolute	4C	3	3	....	
	Indirect	6C	6	3		
	Abs. index indirect	7C	6	3		(4)
JSR	Absolute	20	6	3	....	
LDA	Immediate	A9	2	2	N.....Z.	
	Absolute	AD	4	3		
	Zero page	A5	3	2		
	Indexed indirect	A1	6	2		(2)
	Indirect indexed	B1	5	2		(2)
LDA	Zero pg. index X	B5	4	2		(2)
	Absolute index X	BD	4	3		(2)
	Absolute index Y	B9	4	3		(2)
	Zero pg. indirect	B2	5	2		(2,4)
LDX	Immediate	A2	2	2	N.....Z.	
	Absolute	AE	4	3		
	Zero page	A6	3	2		
	Zero pg. index Y	B6	4	2		(2)
	Absolute index Y	BE	4	3		(2)
LDY	Immediate	A0	2	2	N.....Z.	
	Absolute	AC	4	3		
	Zero page	A4	3	2		
	Zero pg. index X	B4	4	2		(2)
	Absolute index X	BC	4	3		(2)
LSR	Absolute	4E	6	3	N.....ZC	
	Zero page	46	5	2		
	Accumulator	4A	2	1		
	Zero pg. index X	56	6	2		(2)
	Absolute index X	5E	6	3		(2)
NOP	Implied	EA	2	1	....	
ORA	Immediate	09	2	2	N.....Z.	
	Absolute	0D	4	3		
	Zero page	05	3	2		
	Indexed indirect	01	6	2		(2)
	Indirect indexed	11	5	2		(2)
	Zero pg. index X	15	4	2		(2)
	Absolute index X	1D	4	3		(2)
	Absolute index Y	19	4	3		(2)
	Zero pg. indirect	12	5	2		(2,4)
PHA	Implied	48	3	1	....	
PHP	Implied	08	3	1	....	
PHX	Implied	DA	3	1	....	(4)
PHY	Implied	5A	3	1	....	(4)
PLA	Implied	68	4	1	N.....Z.	
PLP	Implied	28	4	1	NV.BDIZC	
PLX	Implied	FA	4	1	N.....Z.	(4)
PLY	Implied	7A	4	1	N.....Z.	(4)
ROL	Absolute	2E	6	3	N.....ZC	
	Zero page	26	5	2		
	Accumulator	2A	2	1		
	Zero pg. index X	36	6	2		(2)
	Absolute index X	3E	6	3		(2)
ROR	Absolute	6E	6	3	N.....ZC	
	Zero page	66	5	2		
	Accumulator	6A	2	1		
	Zero pg. index X	76	6	2		(2)
	Absolute index X	7E	6	3		(2)
RTI	Implied	40	6	1	NV.BDIZC	
RTS	Implied	60	6	1	....	
SBC	Immediate	E9	2	2	NV....ZC	(1)
	Absolute	ED	4	3		(1)
	Zero page	E5	3	2		(1)
	Indexed indirect	E1	6	2		(1,2)
	Indirect indexed	F1	5	2		(1,2)
	Zero pg. index X	F5	4	2		(1,2)
	Absolute index X	FD	4	3		(1,2)
	Absolute index Y	F9	4	3		(1,2)
	Zero pg. indirect	F2	5	2		(1,2,4)
SEC	Implied	38	2	1	C	
SED	Implied	F8	2	1	....D...	
SEI	Implied	78	2	1	.....I..	
STA	Absolute	8D	4	3	....	
	Zero page	85	3	2		
	Indexed indirect	81	6	2		
	Indirect indexed	91	6	2		
	Zero pg. index X	95	4	2		
	Absolute index X	9D	5	3		
	Absolute index Y	99	5	3		
	Zero pg. indirect	92	5	2		(4)
STX	Absolute	8E	4	3	....	
	Zero page	86	3	2		
	Zero pg. index Y	96	4	2		
STY	Absolute	8C	4	3	....	
	Zero page	84	3	2		
	Zero pg. index X	94	4	2		
STZ	Absolute	9C	4	3	....	(4)
	Zero page	64	3	2		(4)
	Zero pg. index X	74	4	2		(4)
	Absolute index X	9E	5	3		(4)
TAX	Implied	AA	2	1	N.....Z.	
TAY	Implied	A8	2	1	N.....Z.	
TRB	Absolute	1C	6	3	......Z.	(4)
	Zero page	14	5	2		(4)
TSB	Absolute	0C	6	3	......Z.	(4)
	Zero page	04	5	2		(4)
TSX	Implied	BA	2	1	N.....Z.	
TXA	Implied	8A	2	1	N.....Z.	
TXS	Implied	9A	2	1	....	
TYA	Implied	98	2	1	N.....Z.	

Notes:

(1) Add one to the number of clock cycles if in decimal mode.

(2) For indexed addressing modes, add one to the number of clock cycles if a page boundary is crossed, i.e., if the index value puts the computed address on a different page than the base address.

(3) Instruction takes two clock cycles if the branch is not taken; three cycles if the branch occurs to the same page; and four if the branch occurs to a different page.

(4) This opcode or addressing mode is available only on the 65C02, not on the 6502.

TABLE 2: Clock Cycles in the Main Loop of CLOCK.TEST

Line No.	Instruction	Current Cycles	Total Cycles
Line 23	LDA, immediate addressing mode	2	—
Line 24	STA, zero-page addressing mode	3	—
Line 25	LDA, immediate addressing mode	2	—
Line 26	STA, zero-page addressing mode	3	—
Line 27	NOP — This is the first instruction of the inner loop.	2	Running total of the inner loop is 2 cycles
Line 28	LDA, zero-page addressing mode	3	5
Line 29	BNE	3 cycles except about once every 256 times through the loop when the branch fails: 3 − 1/256=2.99609375	7.99609375
Line 30	DEC, zero-page addressing mode	Occurs about once every 256 times through loop: 5/256 cycles	8.0155625
Line 31	DEC, zero-page addressing mode	Occurs every time through loop: 5 cycles	13.0155625
Line 32	LDA, zero-page addressing mode	3	16.0155625
Line 33	ORA, zero-page addressing mode	3	16.0155625
Line 34	BNE	3	22.0155625 cycles in the inner loop
Line 35	DEY	2	—
Line 36	BNE	3	—

Clock Cycles

The number of clock cycles required by each of the 6502/65C02 opcodes is given in **Table 1**. In addition, the table gives information on addressing modes allowed for each of the instruction mnemonics, the number of bytes used by each opcode, and the Processor Status flags affected by each mnemonic.

We will now use the clock cycles in **Table 1** to calculate the time required between beeps in CLOCK.TEST. **Lines 16-22** of **Listing 1** are preliminaries to the actual timing loop. We will therefore just go through the main loop, **lines 23-36**, determining the clock cycles of each instruction (see **Table 2**). Once we know the total number of cycles, we can use the Apple's clock rate to determine the total time.

The number of passes through the inner loop of CLOCK.TEST is TIMCNT (= $B511 = 46,353). Therefore, the total number of clock cycles through the inner loop is 46,353 loops × 22.0155625 cycles/loop = 1,020,487 cycles. Since the clock rate of the Apple is 1.02048432 million cycles per second, the time used by the main loop of the program is 1,020,487 cycles divided by 1,020,484.32 cycles per second = 1.0000026 second, or rounded to the nearest 1/100 sec., 1.00 second. (Obviously, the number $B511 = 46353 was selected so the time of the inner loop would come out as exactly one second.)

The other instructions between the CLOCK.TEST beeps yield a total of about 15 cycles, which amounts to only 0.0000147 second — an insignificant time compared to the accuracy of your stopwatch. Since the main loop is executed five times in the program, the total time of the main loop is exactly 5.00 seconds.

Finally, the Apple BELL routine beeps the speaker for 0.10 second. Therefore, the total time from the start of one beep to the start of the second beep is exactly 5.10 seconds, which is what you get from careful stopwatch measurements of CLOCK.TEST.

Using a stopwatch to measure time between beeps may be instructive, but it's hardly exciting.

AMPER.MUSIC

Using a stopwatch to measure time between beeps may be instructive, but it's hardly exciting. A more interesting example of assembly language timing occurs in programming computer music.

Listing 2 gives the source code of a program used to generate sound effects and music within an Applesoft BASIC program. Type the listing into your assembler/editor, assemble the program, and save the source and object codes to disk with the base name AMPER.MUSIC. If you don't have an assembler, you can enter the Monitor with CALL −151 and key in the hex code. Save the program with the command:

```
BSAVE AMPER.MUSIC,A$2E4,L$D7
```

Important note: If you are entering this program from the Monitor, do not try to enter all of the hex values in two or three input lines. The program begins at $2E4, which is at the end of the Apple's input buffer. If you try to enter too much at once, there is a chance that the characters you enter will overwrite the program code in memory. If you enter the bytes in groups of 16 or fewer you should have no problems.

Using AMPER.MUSIC

To use the program, type **BRUN AMPER .MUSIC** from BASIC immediate mode, or within your Applesoft BASIC program, include the statement:

```
PRINT CHR$(4);"BRUN AMPER
.MUSIC"
```

This connects the routine to Applesoft's ampersand (&) hook (at $3F5 through $3F7). From that point on, when BASIC encounters the &, it places the value of the byte immediately following the & into the Accumulator and executes a JSR $3F5. The opcode $4C (JMP) and its two-byte jump address at $3F5, $3F6 and $3F7 redirect program control to the start of the main body of AMPER.MUSIC.

Use the ampersand commands shown in **Table 3** from BASIC.

Note Frequencies and Clock Cycles

AMPER.MUSIC was written so that the duration *d* is independent of the pitch (or frequency) of the note. This is accomplished by making the sound loop (**lines 119-132** in **Listing 2**) take up the same amount of time whether or not the speaker is accessed. During each pass of the loop, the speaker may be accessed to click the speaker, or not accessed to produce no sound. High-pitched

TABLE 3: BASIC Ampersand Commands

Command	Function
&n,d	Plays note number *n* with a duration of *d*. The note number ranges from 0 to 64. Each number (except zero, which is a rest) represents a note of the musical scale; for example, 1 is F# (1½ octaves below middle C), 2 is G, 3 is G#, and so forth up through the half-step (monochromatic or 12-note) scale. Note number 19 is middle C, and 28 is the A-440 (meaning 440 Hz, the pitch on which the standard musical scale is based). After about note number 50, the pitches do not follow the normal 12-note scale. The duration *d* must be a number within the range 0 to 255.
&n	Plays note number *n* using the default (most recently specified) duration. The original default, before a duration has been specified from BASIC, is 100.
&	Plays the default (most recently specified) note using the default duration.
&0,d	Rests (pauses with no sound) for the duration *d*.
&STOP	Stops (ignores) all subsequent ampersand commands. This is a way of turning off the sound from within a BASIC program.
&RESUME	Resumes execution of all subsequent ampersand commands.

notes are produced by clicking the speaker at a high frequency, for example, every ten times through the loop. Low-pitched notes are produced by clicking the speaker at a low frequency, for example, only once every two hundred times through the loop.

In AMPER.MUSIC, the pitch parameter FREQ (which determines the frequency of clicking the speaker) is stored in the X-Register. Each pass through the loop, X is decremented (**line 119**). When X reaches zero, the program branches (from **line 120**) to the instruction that clicks the speaker (in **line 124**), and resets the X-Register to the pitch parameter FREQ. If X is not zero, the branch in **line 120** fails, and **lines 121-123** are executed. These lines (**121-123**) take up the exact same number of clock cycles as when the speaker is clicked.

> ***After a few moments, Bach's Inventio VI will begin to play.***

Lines 129-132 decrement the two-byte value that determines the duration. When both bytes (one in the Y-Register and one in DURCNT+1) reach zero, the loop ends, and the sound terminates.

Go through **lines 119-132** on your own and try to determine the number of clock cycles used when the speaker is accessed and when it is not. You will find that, under either circumstance, the average number of cycles in the loop is 22.0351563.

Here's how to come up with the answer. **Table 4** compares the number of cycles used in executing **lines 119-126** (see **Listing 2**) with their respective number of cycles, when the speaker is *not* clicked and when the speaker *is* clicked.

Lines 127-132 are a little more complicated. **Line 127**: 2.99609375 (three cycles when the branch is taken and two when it is not, which happens once every 256 cycles, for an average of 3 − 1/256 = 2.9960938 cycles); **line 128**: 0.01953125 (five cycles taken once every 256 times through the loop, or 5/256 = 0.01953125); **line 129**: 2; **line 130**: 2.99609375 (three cycles when the branch is taken and two when it is not); **line 131**: 0.01171875 (three when the above branch is not taken, or 3/256 = 0.0117188); **line 132**: 0.01171875 (the total for **lines 127-132** is 8.03515625). The total for the two parts of the main sound loop is 14 + 8.03515625 = 22.03515625.

PITCH.CALC

You can calculate the required parameter (for the PITCHTAB in AMPER.MUSIC) given the note frequency, using the following formula:

```
PITCH = 1,020,484 Hz/(2 *
22.03515625 * frequency)
```

PITCH.CALC (**Listing 3**) does this computation (see **line 180**) using the frequency of all notes from F# with a frequency of 87.3 (1½ octaves below middle C), up to C with a frequency of 2093 Hz (3 octaves above middle C). The two in the denominator of the above equation is necessary because only half of the accesses of the Apple speaker cause a sound (moving the speaker diaphragm out); the other half move the diaphragm in (without making a sound).

PITCH.CALC prints the note numbers 0 to 55 and their corresponding note names

TABLE 4: Cycle Comparison for Lines 119-126

Line Number	No Click	With Click	
119	2	2	
120	2	3	Add 1 for branch taken
121	3	—	
122	2	—	
123	3	—	Branch always taken
124	—	4	
125	—	3	
126	2	2	
	14	14	

(C, C#, G, G#, etc.), frequencies, and PITCH values used in AMPER.MUSIC. The following formula is used to derive the frequency of each note N:

```
Frequency = 87.3078 •
(1.059463093)C^N
```

Notes 56-64 are allowed in AMPER .MUSIC but give nonstandard notes; therefore, they should not be used in music but only in special sound effects. You will want to get a printout of the notes using PITCH .CALC as an aid in using AMPER.MUSIC in your Applesoft programs.

The pitches of higher notes are less accurate . . .

The PITCH values obtained from PITCH .CALC make up the pitch table (PITCHTAB) in **lines 141-146** of **Listing 1**. This data yields fairly accurate note pitches, especially for the lower-numbered notes. The pitches of higher notes are less accurate because the PITCH parameter has to be an integer (you can't click the speaker every 12.5 times through the sound loop; it has to be 12 or 13). If the clock rate of the microprocessor were greater, you could get a wider range of accurate notes using the same algorithm.

BACH

Listing 4 (BACH) demonstrates the use of AMPER.MUSIC in an Applesoft BASIC program. Enter **Listing 4** into your Apple, save the program to disk, and type RUN.

After a few moments, Bach's Inventio VI will begin to play. Use these keyboard commands to modify the music:

- Press Escape to quit BACH
- Press keys 1-9 to adjust the tempo (1 is low; 9 is high)
- Press S to toggle sound on/off

How BACH Works

Memory location −16384 ($C000) is the keyboard soft-switch buffer. It contains the ASCII code of any key that the user presses. If the ASCII code there is less than 128 (the first of the control characters), the user has not pressed a key since the keyboard strobe was cleared (i.e., since the last access of memory location −16368 = $C010).

In BACH, if you haven't pressed a key, the program jumps to **line 220**, where it calculates the next note index and (in **line 230**) plays the note. If you press a key, BACH assigns the ASCII code to the variable K (**line 150**). If the key is Escape, K = 155 and the program terminates. If the key is S, K = S and the sound flag SF toggles between zero and 1; if SF is zero, & STOP terminates all & notes; if SF is 1, & RESUME reactivates the & notes. If the key you press is a number 1 through 9, BACH calculates the delay factor DF (see **line 210**) based upon your numeric input.

SUMMARY

The average clock rate of the Apple II is 1.02048432 MHz (megahertz, or million cycles per second). Each machine language opcode requires a certain number of clock cycles, as shown in **Table 1**. Having this data available allows you to accurately calculate the time expended by any (or part of any) machine language routine.

Table 1 also serves as a review of the 6502/65C02 mnemonics, most of which we have discussed in Nibbling at Assembly Language.

Listing 3 for DOS Device Detective
DETECTIVE.DEMO

```
10  REM *********************
20  REM *   DETECTIVE.DEMO   *
30  REM *  BY JOHN R. VOKEY  *
40  REM * COPYRIGHT (C) 1987 *
50  REM * BY MICROSPARC, INC *
60  REM * CONCORD, MA  01742 *
70  REM *********************
80  REM DISPLAY TITLE PAGE
90  PRINT  CHR$ (14); CHR$ (21): HOME :DRIVE =
    43624:SLOT = DRIVE + 2
100  COLOR= 2: GOSUB 470
110  POKE 33,38: POKE 32,1: POKE 34,1: POKE 3
    5,23
120  FOR I = 5 TO 21: READ S$
130  FOR J = 23 TO I STEP  - 1
140  VTAB J: GOSUB 490
150  NEXT : NEXT
160  DATA    DOS DEVICE DETECTIVE,DEVICE-INDE
    PENDENT DOS,BY JOHN VOKEY,.......,COPYRIG
    HT (C) 1987
170  DATA  MICROSPARC INC.
180  DATA  CONCORD MA 01742
190  DATA  ....
200  REM INSTALL PATCH
210  PRINT : PRINT  CHR$ (4)"BRUN DETECTIVE,A
    $2000"
220  VTAB 10: HTAB 12: PRINT "<PATCH INSTALLE
    D>"
230  REM DELAY FOR 1000 OR KEY
240  VTAB 24: HTAB 15: INVERSE
250  PRINT "PRESS <RETURN>";: NORMAL : POKE  -
    16368,0: FOR I = 1 TO 500: IF  PEEK ( -
    16384) < 128 THEN  NEXT
260  REM DISPLAY INSTRUCTIONS
270  VTAB 7: CALL  - 958: FOR I = 9 TO 12: READ
    S$: FOR J = 23 TO I STEP  - 1: VTAB J
280  GOSUB 490
290  NEXT : NEXT : VTAB 24: HTAB 15: INVERSE
    : PRINT "              ";: NORMAL : REM
     14 SPACES
300  DATA  PLEASE INSERT THE DETECTIVE DISK
310  DATA  INTO ANY DRIVE ON THE SYSTEM,(OR N
    OT AT ALL!)
320  DATA THEN PRESS <RETURN>
330  ONERR  GOTO 510
340  POKE  - 16368,0
350  REM AWAIT KEYPRESS
360  VTAB 13: HTAB 19: GET S$: IF S$ <  >  CHR$
    (13) AND S$ <  >  CHR$ (27) THEN 360
370  IF S$ =  CHR$ (27) THEN 450
380  REM SEARCH FOR FILE
390  PRINT : IF  NOT ERR THEN  PRINT  CHR$ (4
    )"VERIFY DETECTIVE"
400  IF ERR THEN  VTAB 20: HTAB 6: PRINT  CHR$
    (7);"DETECTIVE IS NOT ON THE SYSTEM"
410  IF  NOT ERR THEN  VTAB 20: HTAB 6: PRINT
     CHR$ (7);"DETECTIVE IS IN SLOT " PEEK (
    SLOT)", DRIVE " PEEK (DRIVE)
420 ERR = 0: VTAB 24: HTAB 15: INVERSE : PRINT
    "<ESC> TO EXIT ";: NORMAL
430  GOTO 360
440  REM EXIT
450  POKE  - 16368,0: TEXT : HOME : POKE 216,
    0: END
460  REM FRAME SUBROUTINE
470  HLIN 0,39 AT 1: FOR K = 1 TO 47 STEP 2: PLOT
    0,K: PLOT 39,K: NEXT : HLIN 0,39 AT 47: RETURN

480  REM PRINT SUBROUTINE
490  HTAB (41 -  LEN (S$)) / 2: PRINT S$;: CALL
     - 958: RETURN
500  REM ON ERR TRAP
510 ERR =  PEEK (222): RESUME
```

END OF LISTING 3

A Matter of Timing

Article on page 70

Listing 1 for A Matter of Timing
CLOCK.TEST

```
                   ;
                   ***********************************************
                   *                CLOCK.TEST                   *
                   *          by S. Scott Zimmerman              *
                   *            Copyright (c) 1987               *
                   *            by MicroSPARC, Inc               *
                   *            Concord, MA  01742               *
                   ***********************************************
                   * MicroSPARC Assembler 3.0
                            ORG $300
                   TINDX    EQU $19          ;Time loop index
                   WAIT     EQU $FCA8        ;Pause accum amount
                   BELL     EQU $FF3A        ;Beep routine

                   TIMCNT   EQU $B511        ;Inner loop time count

0300  A0 0A                 LDY #10          ;Make a pause before
0302  A9 FF        PAUSLOOP LDA #$FF         ; starting test
0304  20 A8 FC              JSR WAIT         ;Go wait a while
0307  88                    DEY              ;End of pause loop?
0308  10 F8                 BPL PAUSLOOP     ;No, go loop again
030A  20 3A FF              JSR BELL         ;Yes, sound start beep
030D  A0 05                 LDY #5           ;Set index for "seconds"
030F  A9 11        SECLOOP  LDA #TIMCNT      ;Set the time loop index
0311  85 19                 STA TINDX        ; to TIMCNT
0313  A9 B5                 LDA #TIMCNT/
0315  85 1A                 STA TINDX+1
0317  EA           TIMELOOP NOP              ;Empty loop
0318  A5 19                 LDA TINDX        ;Do a 16-bit (double
031A  D0 02                 BNE DECINDX      ; precision) decrement
031C  C6 1A                 DEC TINDX+1      ;Dec HOB as needed
031E  C6 19        DECINDX  DEC TINDX        ;Always dec LOB
0320  A5 19                 LDA TINDX        ;Is the time index zero?
0322  05 1A                 ORA TINDX+1
0324  D0 F1                 BNE TIMELOOP     ;No, so loop again
0326  88                    DEY              ;Yes. End of "sec" loop?
0327  D0 E6                 BNE SECLOOP      ;No, loop again
0329  4C 3A FF              JMP BELL         ;Yes, so do end beep

000  Errors

0300  Hex Start of Object
032B  Hex end of Object
002C  Hex Length of Object
7B86  Hex end of Symbols
```

END OF LISTING 1

Listing 2 for A Matter of Timing
AMPER.MUSIC

```
                   ;
                   ***********************************************
                   *                                             *
                   *                AMPER.MUSIC                  *
                   *                                             *
                   *          by S. Scott Zimmerman              *
                   *            Copyright (c) 1987               *
                   *            by MicroSPARC, Inc               *
                   *            Concord, MA  01742               *
                   *          MicroSPARC Assembler 3.0           *
                   ***********************************************

                            ORG $2E4         ;Start in input buffer

                   ***********************************************
                   *  Zero-page EQUates and constants:           *
                   ***********************************************

                   PITCH    EQU $19          ;Pitch parameter
                   DURATION EQU $1A          ;Duration parameter
                   FREQ     EQU $1B          ;Frequency
                   DURCNT   EQU $1C          ;Duration count, snd loop
                   STOPFLG  EQU $1E          ;Stop flag for no sound

                   RESUMTOK EQU $A6          ;Applesoft RESUME token
                   STOPTOK  EQU $B3          ;Applesoft STOP token
```


```
                            ORG $2E4         ;Start in input buffer

                   ***********************************************
                   *  Zero-page EQUates and constants:           *
                   ***********************************************

                   PITCH    EQU $19          ;Pitch parameter
                   DURATION EQU $1A          ;Duration parameter
                   FREQ     EQU $1B          ;Frequency
                   DURCNT   EQU $1C          ;Duration count, snd loop
                   STOPFLG  EQU $1E          ;Stop flag for no sound

                   RESUMTOK EQU $A6          ;Applesoft RESUME token
                   STOPTOK  EQU $B3          ;Applesoft STOP token
```

```
                           *********************************************
                           *  Monitor locations and routines:          *
                           *********************************************

                           CHRGOT   EQU $B7          ;Get text character
                           AMPER    EQU $3F5         ;&-Routine address
                           ERROR    EQU $D412        ;Applesoft error handler
                           TXTEND   EQU $D995        ;Move TXTPTR to end
                           GETBYTC  EQU $E6F5        ;Eval text expr w/ comma
                           GETBYT   EQU $E6F8        ;Eval text expr no comma
                           SPEAKER  EQU $C030        ;Speaker soft switch

                           *********************************************
                           *  Set up ampersand vector and parm defaults: *
                           *********************************************

02E4  A9 4C                         LDA #$4C         ;Get JMP opcode
02E6  8D F5 03                      STA AMPER        ;Put in &-vector
02E9  A9 00                         LDA #START       ;Set amper vector to
02EB  8D F6 03                      STA AMPER+1      ; the START address
02EE  A9 03                         LDA #START/      ;Do the HOB
02F0  8D F7 03                      STA AMPER+2
02F3  A9 13                         LDA #19          ;Set default pitch to
02F5  85 19                         STA PITCH        ; middle C
02F7  A9 64                         LDA #100         ;Set default duration
02F9  85 1A                         STA DURATION
02FB  A9 00                         LDA #0           ;Clear stop flag for
02FD  85 1E                         STA STOPFLG      ; speaker on default
02FF  60                            RTS              ;End of setup

                           *********************************************
                           *  Start of main program:                   *
                           *********************************************

0300  C9 B3                START    CMP #STOPTOK     ;Is STOP token there?
0302  D0 07                         BNE CHKRESUM     ;No, go check for RESUME
0304  A9 01                         LDA #1           ;Yes, set stop flag
0306  85 1E                         STA STOPFLG      ; so no sound is made
0308  4C 95 D9                      JMP TXTEND       ;Exit to end of command

030B  C9 A6                CHKRESUM CMP #RESUMTOK    ;Is RESUME token there?
030D  D0 07                         BNE CHKSTOP      ;No, go check stop flag
030F  A9 00                         LDA #0           ;Yes, clear stop flag
0311  85 1E                         STA STOPFLG      ; so sound is made
0313  4C 95 D9                      JMP TXTEND       ;Exit to end of command

0316  A5 1E                CHKSTOP  LDA STOPFLG      ;Is stop flag set?
0318  F0 03                         BEQ GETPARM      ;No, go get parameters
031A  4C 95 D9                      JMP TXTEND       ;Yes, ignore command

031D  20 B7 00             GETPARM  JSR CHRGOT       ;Is first parm there?
0320  F0 0F                         BEQ SNDDET       ;No, go make sound
0322  20 F8 E6                      JSR GETBYT       ;Yes, get pitch parm
0325  86 19                         STX PITCH        ;Save it

0327  20 B7 00                      JSR CHRGOT       ;Is second parm there?
032A  F0 05                         BEQ SNDDET       ;No, go make sound
032C  20 F5 E6                      JSR GETBYTC      ;Yes, get duration parm
032F  86 1A                         STX DURATION     ;Save it

                           *********************************************
                           *  Sound parameter determinations:          *
                           *********************************************

0331  A6 19                SNDDET   LDX PITCH        ;Get current pitch
0333  E0 41                         CPX #65          ;Is it 64 or less?
0335  90 05                         BCC OKAY         ;Yes, value is okay
0337  A2 35                         LDX #53          ;Get ILLEGAL QUANTITY
0339  4C 12 D4                      JMP ERROR        ; error and print it

033C  BD 7A 03             OKAY     LDA PITCHTAB,X   ;Get frequency
033F  85 1B                         STA FREQ         ;Save it
0341  D0 03                         BNE DODUR        ;Not zero, do duration
0343  8D 64 03                      STA SPKR+1       ;Clear speaker for rest

0346  A5 1A                DODUR    LDA DURATION     ;Put the duration into
0348  85 1C                         STA DURCNT       ; temporary variable
034A  A9 00                         LDA #0           ;Zero the HOBs
034C  85 1D                         STA DURCNT+1

034E  A2 08                         LDX #8           ;Multiply by 256
0350  06 1C                MULTLOOP ASL DURCNT       ;Shift left to multiply
0352  26 1D                         ROL DURCNT+1     ; 16-bit number by 2
0354  CA                            DEX              ;End of loop?
0355  D0 F9                         BNE MULTLOOP     ;No, go again
0357  A4 1C                         LDY DURCNT       ;Put LOB in register
0359  A6 19                         LDX PITCH        ;Initialize pitch index

                           *********************************************
                           *  Sound loop:                              *
                           *********************************************

035B  CA                   SNDLOOP  DEX              ;Is pitch index zero?
035C  F0 05                         BEQ SPKR         ;Yes, so click speaker
035E  24 1B                         BIT FREQ         ;Stall 3 cycles
0360  B8                            CLV              ;Stall/force branch
0361  50 05                         BVC DECDUR       ;(Always) skip click
0363  2C 30 C0             SPKR     BIT SPEAKER      ;Click spkr (unless rest)
0366  A6 1B                         LDX FREQ         ;Restore frequency to
0368  98                   DECDUR   TYA              ;16-bit decrement
0369  D0 02                         BNE DECLOB       ;Do LOB only
036B  C6 1D                         DEC DURCNT+1     ;Decrement HOB
036D  88                   DECLOB   DEY              ;Decrement LOB
036E  D0 EB                         BNE SNDLOOP      ;Go again if not zero
0370  A5 1D                         LDA DURCNT+1     ;Is HOB zero
0372  D0 E7                         BNE SNDLOOP      ;No, go thru sound loop
0374  A9 30                         LDA #SPEAKER     ;Restore speaker in
0376  8D 64 03                      STA SPKR+1       ; case it was a rest
0379  60                            RTS              ;End of AMPER.MUSIC

                           *********************************************
                           *  Pitch table:                             *
                           *********************************************

037A  00 FA EC             PITCHTAB DFC 0,250,236,223,211,199,188,177,167
      DF D3 C7 BC B1 A7
0383  9E 95 8C                      DFC 158,149,140,133,125,118,112,105,99
      85 7D 76 70 69 63
038C  5E 59 54                      DFC 94,89,84,79,74,70,66,63,59,56,53
      4F 4A 46 42 3F 3B
      38 35
0397  32 2F 2C                      DFC 50,47,44,42,39,37,35,33,31,30,28
      2A 27 25 23 21 1F
      1E 1C
03A2  1A 19 17                      DFC 26,25,23,22,21,20,19,18,17,16,15
      16 15 14 13 12 11
      10 0F
03AD  0E 0D 0C                      DFC 14,13,12,11,10,9,8,7,6,5,4,3,2,1
      0B 0A 09 08 07 06
      05 04 03 02 01

000  Errors

02E4  Hex Start of Object
03BA  Hex end of Object
00D7  Hex Length of Object
7806  Hex end of Symbols
```

END OF LISTING 2

Listing 3 for A Matter of Timing

PITCH.CALC

```
10  REM **********************
20  REM *     PITCH.CALC     *
30  REM * BY SCOTT ZIMMERMAN *
40  REM * COPYRIGHT (C) 1987 *
50  REM * BY MICROSPARC, INC *
60  REM * CONCORD, MA  01742 *
70  REM **********************
80  REM
90  DIM N$(12): GOSUB 300
100  FOR I = 1 TO 12: READ N$(I): NEXT I
110  VTAB 20: CALL  - 958: PRINT "SEND OUTPUT
      TO PRINTER? (Y/N) ";: GET A$: PRINT A$
120  IF A$ = "Y" OR A$ = "y" THEN  PRINT  CHR$
     (4);" PR# 1": GOTO 130
130 HZ = 1.02048E6: REM APPLE FREQUENCY
140 CL = 2: REM CLICK FACTOR (CLICK ONCE EVER
     Y TWO ACCESSES OF SPEAKER)
150 CY = 22.0352: REM CYCLES PER LOOP IN AMPE
     R.MUSIC
160  HOME : PRINT "NOTE  NOTE #   NOTE FREQ
     PITCH PARM": POKE 34,2: HOME : FOR N = 0
      TO 55
170 FR = 87.3079 * (1.059463093) ^ N: REM CAL
     C NOTE FREQUENCY
180 PITCH = HZ / (CL * CY * FR)
190 PITCH =  INT (PITCH + .5): REM ROUND TO N
     EAREST INTEGER
200 F$ =  STR$ ( INT ((FR) * 100)): REM ROUND
      OFF TO NEAREST HUNDREDTH
210 L =  LEN (F$) - 2:F$ =  LEFT$ (F$,L) + ".
     " +  RIGHT$ (F$,2)
220  GOSUB 320: HTAB 8: PRINT N;: HTAB 18: PRINT
     F$;: HTAB 29: PRINT PITCH
230  IF  PEEK ( - 16384) < 128 THEN 260
240  GET A$: IF A$ =  CHR$ (27) THEN 290
250  POKE  - 16368,0: GET A$:
260  NEXT N
270  PRINT  CHR$ (4);"PR# 0"
280  PRINT "SEE THEM AGAIN? ";: GET A$: IF A$
      = "Y" THEN  GOSUB 300: GOTO 110
290  TEXT : END
300  TEXT : HOME : VTAB 6: HTAB 13: INVERSE :
      PRINT " PITCH CALC ": NORMAL
310  VTAB 8: HTAB 10: PRINT "BY SCOTT ZIMMERM
     AN": HTAB 10: PRINT "COPYRIGHT (C) 1987"
     : HTAB 10: PRINT "BY MICROSPARC, INC": RETURN

320  IF N = 0 THEN  PRINT "REST";:PITCH = 0: RETURN

330  IF  INT (FR) = 261 THEN  HTAB 2: PRINT "
     C MID";: RETURN
340 B = N + 9:A = B - 12 *  INT ((B - 1) / 12
     ): HTAB 2: PRINT N$(A);: RETURN
350  DATA A,A#,B,C,C#,D,D#,E,F,F#,G,G#
```

END OF LISTING 3

Listing 4 for A Matter of Timing

BACH

```
10  REM ***********************
20  REM *        BACH         *
30  REM * BY SCOTT ZIMMERMAN  *
40  REM * COPYRIGHT (C) 1987  *
50  REM * BY MICROSPARC, INC  *
60  REM * CONCORD, MA  01742  *
70  REM ***********************
80  REM
90  TEXT : HOME : GOSUB 250
100  IF  PEEK (1015) <  > 3 THEN  PRINT  CHR$
     (4);"BRUN AMPER.MUSIC"
110 N = 103: DIM P(N),D(N)
120  FOR I = 1 TO N: READ P(I): NEXT
130  FOR I = 1 TO N: READ D(I): NEXT
140 NN = 0:DF = 2.5:SF = 1
150 K =  PEEK ( - 16384): IF K < 128 THEN 220
160  POKE  - 16368,0
170  IF K = 155 THEN  TEXT : HOME : END
180  IF K = 211 THEN  &  STOP :SF =  NOT SF: IF
     SF THEN  &  RESUME
190  IF K < 177 THEN 220
200  IF K > 185 THEN 220
210 DF = (K - 176) / 2
220 NN = NN + 1: IF NN > N THEN  & 0,255:NN =
     1
230  & P(NN),D(NN) * DF
240  GOTO 150
250  VTAB 5: INVERSE :A$ = " BACH ": GOSUB 28
     0: NORMAL
260  VTAB 7:A$ = "PROGRAMMED WITH AMPER.MUSIC
     ": GOSUB 280: PRINT
270 A$ = "BY SCOTT ZIMMERMAN": GOSUB 280:A$ =
     "COPYRIGHT (C) 1987": GOSUB 280:A$ = "BY
      MICROSPARC, INC": GOSUB 280: RETURN
280  HTAB (41 -  LEN (A$)) / 2: PRINT A$: RETURN

290  DATA 24,28,24,31,24,36,35,33,31,33,31,29
     ,28,29,28,26,24,28,31,28,36,31,40,43,41,
     43,40,43,41,43,40,43,41,43,36,40,38,40,3
     6,40,38,40,36,40,38,40
300  DATA 33,36,35,36,33,36,35,36,33,36,35,36
     ,30,26,33,30,36,33,38,40,38,36,35,36,35,
     33,31,33,31,29,28,33,31,30,31,30,28,26,2
     8,26,24,23,24,23,21,19,31,30,31,23,24,31
     ,23,31,21,30,31
310  DATA 18,18,18,18,18,18,8,8,8,8,8,8,8,8,8
     ,8,18,18,18,18,18,18,8,8,8,8,8,8,8,8,8,8
     ,8,8,8,8,8,8,8,8,8,8,8,8,8,8
320  DATA 8,8,8,8,8,8,8,8,8,8,8,8,18,18,18,18
     ,18,18,8,8,8,8,8,8,8,8,8,8,8,8,18,8,8,8,
     8,8,8,8,8,8,8,8,8,8,8,18,8,8,18,18,18,18
     ,18,18,18,18,38
```

END OF LISTING 4

THE ERROR TRAP

Arcade Sound Editor (Vol. 8/No. 1, p. 35): The duplicate code near the end of Listing 2 (DUO) after "End Assembly, 445 bytes, Errors: 0" need not by typed.