Listing 2, continued:

```
0876: DC        00098          DFB  DCR-1     (FX)
0877: 5E        00099          DFB  MUL-1     (E)
0878: 5E        00100          DFB  MUL-1     (UNUSED)
0879: 5E        00101          DFB  MUL-1     (F)
087A: 10 CA     00102 SET      BPL  SET1      ALWAYS TAKEN
087C: B5 00     00103 LD       LDA  R0L,X
                00104 BK       ECU  *-1
087E: 85 00     00105          STA  R0L
0880: B5 01     00106          LDA  R0H,X     MOVE RX TO R0
0882: 85 01     00107          STA  R0H
0884: 60        00108          RTS
0885: A5 00     00109 ST       LDA  R0L
0887: 95 00     00110          STA  R0L,X     MOVE R0 TO RX
0889: A5 01     00111          LDA  R0H
088B: 95 01     00112          STA  R0H,X
088D: 60        00113          RTS
088E: A5 00     00114 STAT     LDA  R0L
0890: 81 00     00115 STAT2    STA  (R0L,X)   STORE BYTE INDIRECT
0892: A0 00     00116          LDY  #$0
0894: 84 1D     00117 STAT3    STY  R14H      INDICATE R0 IS RESULT REG
0896: F6 00     00118 INR      INC  R0L,X
0898: D0 02     00119          BNE  INR2      INCR RX
089A: F6 01     00120          INC  R0H,X
089C: 60        00121 INR2     RTS
089D: A1 00     00122 LDAT     LDA  (R0L,X)   LOAD INDIRECT (RX)
089F: 85 00     00123          STA  R0L       TO R0
08A1: A0 00     00124          LDY  #$0       ZERO HIGH-ORDER R0 BYTE
08A3: 84 01     00125          STY  R0H
08A5: F0 ED     00126          BEQ  STAT3     ALWAYS TAKEN
08A7: A0 00     00127 POP      LDY  #$0       HIGH ORDER BYTE = 0
08A9: F0 06     00128          BEQ  POP2      ALWAYS TAKEN
08AB: 20 DD 08  00129 POPD     JSR  DCR       DECR RX
08AE: A1 00     00130          LDA  (R0L,X)   POP HIGH-ORDER BYTE FROM RX
08B0: A8        00131          TAY            SAVE IN Y-REG
08B1: 20 DD 08  00132 POP2     JSR  DCR       DECR RX
08B4: A1 00     00133          LDA  (R0L,X)   LOW-ORDER BYTE
08B6: 85 00     00134          STA  R0L       TO R0
08B8: 84 01     00135          STY  R0H
08BA: A0 00     00136 POP3     LDY  #$0       INDICATE R0 AS LAST
08BC: 84 1D     00137          STY  R14H            RESULT REG
08BE: 60        00138          RTS
08BF: 20 9D 08  00139 LDDAT    JSR  LDAT      LOW BYTE TO R0, INCR RX
08C2: A1 00     00140          LDA  (R0L,X)   HIGH-ORDER BYTE TO R0
08C4: 85 01     00141          STA  R0H
08C6: 4C 96 08  00142          JMP  INR       INCR RX
08C9: 20 8E 08  00143 STDAT    JSR  STAT      STORE INDIRECT LOW-ORDER
08CC: A5 01     00144          LDA  R0H       BYTE AND INCR RX.  THEN
08CE: 81 00     00145          STA  (R0L,X)   STORE HIGH-ORDER BYTE.
08D0: 4C 96 08  00146          JMP  INR       INCR RX AND RETURN
08D3: 20 DD 08  00147 STPAT    JSR  DCR       DECR RX
08D6: A5 00     00148          LDA  R0L
08D8: 81 00     00149          STA  (R0L,X)   STORE R0 LOW BYTE FROM RX
08DA: 4C DA 08  00150          JMP  POP3      INDICATE R0 AS LAST
08DD: B5 00     00151 DCR      LDA  R0L,X           RESULT REG
08DF: D0 02     00152          BNE  DCR2      DECR RX
08E1: D6 01     00153          DEC  R0H,X
08E3: D6 00     00154 DCR2     DEC  R0L,X
08E5: 60        00155          RTS
08E6: A0 00     00156 SUB      LDY  #$0       RESULT TO R0
08E8: 38        00157 CPR      SEC            NOTE Y-REG + 1D=2 FOR CPR
08E9: A5 00     00158          LDA  R0L
08EB: F5 00     00159          SBC  R0L,X
08ED: 99 00 00  00160          STA  R0L,Y     R0-RX TO RY
08F0: A5 01     00161          LDA  R0H
08F2: F5 01     00162          SBC  R0H,X
08F4: 99 01 00  00163 SUB2     STA  R0H,Y
08F7: 98        00164          TYA            LAST RESULT REG+2
08F8: 69 02     00165          ADC  #$0       CARRY TO LSB
08FA: 85 1D     00166          STA  R14H
08FC: 60        00167          RTS
08FD: A5 00     00168 ADD      LDA  R0L
08FF: 75 00     00169          ADC  R0L,X
0901: 85 00     00170          STA  R0L       R0+RX TO R0
0903: A5 01     00171          LDA  R0H
0905: 75 01     00172          ADC  R0H,X
0907: A0 00     00173          LDY  #$0       R0 FOR RESULT
0909: F0 09     00174          BEQ  SUB2      FINISH ADD
090B: A6 1E     00175 DS       LDX  R15L      NOTE X-REG IS 1B+2!
090D: 20 90 08  00176          JSR  STAT2     PUSH LOW PC BYTE VIA R12
0910: A5 1F     00177          LDA  R15H
0912: 20 90 08  00178          JSR  STAT2     PUSH HIGH-ORDER PC BYTE
0915: 18        00179 BP       CLC
0916: B0 0E     00180 BNC      BCS  BNC2      NO CARRY TEST
0918: B1 1E     00181 BR1      LDA  (R15L),Y  DISPLACEMENT BYTE
091A: 10 01     00182          BPL  BR2
091C: 88        00183          DEY
091D: 65 1E     00184 BR2      ADC  R15L      ADD TO PC
091F: 85 1E     00185          STA  R15L
0921: 98        00186          TYA
0922: 65 1F     00187          ADC  R15H
0924: 85 1F     00188          STA  R15H
0926: 60        00189 BNC2     RTS
0927: B0 EC     00190 BC       BCS  BR1
0929: 60        00191          RTS
092A: 0A        00192 DP       ASL  A         DOUBLE RESULT-REG INDEX
092B: AA        00193          TAX            TO X-REG FOR INDEXING
092C: B5 01     00194          LDA  R0H,X     TEST FOR PLUS
092E: 10 E8     00195          BPL  BR1       BRANCH IF SO
0930: 60        00196          RTS
0931: 0A        00197 DM       ASL  A         DOUBLE RESULT-REG INDEX
0932: AA        00198          TAX
0933: B5 01     00199          LDA  R0H,X     TEST FOR MINUS
0935: 30 E1     00200          BMI  BR1
0937: 60        00201          RTS
0938: 0A        00202 DE       ASL  A         DOUBLE RESULT-REG INDEX
0939: AA        00203          TAX
093A: B5 00     00204          LDA  R0L,X     TEST FOR ZERO
093C: 15 01     00205          ORA  R0H,X     (BOTH BYTES)
093E: F0 D8     00206          BEQ  BR1       BRANCH IF SO
0940: 60        00207          RTS
0941: 0A        00208 DMZ      ASL  A         DOUBLE RESULT-REG INDEX
0942: AA        00209          TAX
0943: B5 00     00210          LDA  R0L,X     TEST FOR NONZERO
0945: 15 01     00211          ORA  R0H,X     (BOTH BYTES)
0947: D0 CF     00212          BNE  BR1       BRANCH IF SO
0949: 60        00213          RTS
094A: 0A        00214 BM1      ASL  A         DOUBLE RESULT-REG INDEX
094B: AA        00215          TAX
094C: B5 00     00216          LDA  R0L,X     CHECK BOTH BYTES
094E: 35 01     00217          AND  R0H,X     FOR AFF (MINUS 1)
```

memory locations like 6502 instructions. The main loop at SW16B repeatedly calls the "execute instruction" routine at SW16C which examines one op code for type and branches to the appropriate subroutine to execute it.

Subroutine SW16C increments the program counter (R15) and fetches the next op code which is either a register operation of the form OP REG (2 hexadecimal digits) with OP between hexadecimal 1 and F, or a nonregister operation of the form 0 OP with OP between hexadecimal 0 and D. Assuming a register operation, the register specification is doubled to account for the 2 byte SWEET16 registers and placed in the X register for indexing. Then the instruction type is determined. Register operations place the doubled register specification in the high order byte of R14 indicating the "prior result register" to subsequent branch instructions. Nonregister operations treat the register specification (right-hand half-byte) as their op code, increment the SWEET16 PC to point at the displacement byte of branch instructions, load the A-Reg with the "prior result register" index for branch condition testing, and clear the Y-Reg.

When Is an RTS Really a JSR?

Each instruction type has a corresponding subroutine. The subroutine entry points are stored in a table which is directly indexed by the op code. By assigning all the entries to a common page, only a single byte of address need be stored per routine. The 6502 indirect jump might have been used as follows to transfer control to the appropriate subroutine:

```
LDA #ADRH     High order address byte
STA IND+1
LDA OPTBL,X   Low order byte
STA IND
JMP (IND)
```

To save code the subroutine entry address (minus 1) is pushed onto the stack, high order byte first. A 6502 RTS (ReTurn from Subroutine) is used to pop the address off the stack and into the 6502 program counter (after incrementing by 1). The net result is that the desired subroutine is reached by executing a subroutine return instruction! [*This ironic situation is an example of what is commonly referred to as "cleverness."*]

Op Code Subroutines

The register operation routines make use of the 6502 "zero page indexed by X" and "indexed by X indirect" addressing modes to access the specified registers and indirect data. The "result" of most register ops is left