A NOTE ON 6502 INDIRECT ADDRESSING

BY W. D. MAURER
George Washington University
S.E.A.S.
Washington, D.C. 20052

There are two kinds of indirect addressing on the 6502, both of them with indexing. Except for JMP, however, indirect addressing *without* indexing is not provided. Since indirect addressing seems to be more useful without indexing than with it, many people have erroneously regarded this as a design flaw.

In fact, anything you can do with indirect addressing without indexing can be done, and done faster, with post-indexed (Y-register) indirect addressing on the 6502. We shall now give several illustrations of this.

Suppose we have two zero-page locations called IA and IA+1 for holding an indirect address. We can initialize these by setting them to AD and AD+1, as follows:

```
LDY  AD
STY  IA
LDY  AD+1
STY  IA+1
```

At this point we may do a JMP (IA), but not a STA (IA) (for example), since that instruction does not exist on the 6502. It may be simulated, of course, by loading the Y register with zero, so that the full sequence becomes

```
LDY  AD
STY  IA
LDY  AD+1
STY  IA+1
LDY  #0
STA  (IA),Y
```

But this is unnecessary. All we have to do is keep IA (the low-order address) set to zero at all times, and write

```
LDY  AD+1
STY  IA+1
LDY  AD
STA  (IA),Y
```

which is a *decrease* in the number of instructions (from 5 to 4), rather than an *increase* (from 5 to 6).

Why does this work? Suppose that the indirect address is 2345. Then AD contains 45 (hex) and AD+1 contains 23. Instead of putting 45 in IA and 23 in IA+1 (giving an indirect address of 2345, without indexing), we put *zero* in IA and 23 in IA+1, giving an indirect address of 2300. To this we apply indexing, with 45 (hex) in the Y register; 2300 plus 45 makes 2345.

A common use of this capability is in processing arrays of strings. Suppose we have an array T of n strings, with $n \leqslant 128$. This is implemented as an array of n addresses, with T($2k-1$) and T($2k$) containing, respectively, the low-order and high-order parts of the address of the first character of the k-th string in the string array T, for $1 \leqslant k \leqslant n$. To get the first character of the k-th string, we perform

```
LDA  K
ASL
TAY
LDA  T-1,Y
STA  IA+1
LDA  T-2,Y
TAY
LDA  (IA),Y
```

This assumes that T(1) is contained at T, T(2) at T+1, etc., so that T($2k-1$) is kept at T+$2k$-2 (= (T-2)+$2k$), and that IA contains the constant zero as before. This time the saving is not quite as great because we cannot do an LDY T-2,Y directly (we could not even do an LDY T-2,X unless T is in page zero). But TAY is certainly both shorter and faster than STA IA (which is what we would have done if indirect addressing without indexing had been available). (Note: ASL, in the above code, shifts the A register left by one, thus multiplying it by 2. On some assemblers one writes ASL A instead of simply ASL for this.)

The same sort of savings accrue if T is a parallel array rather than a serial array. That is, suppose we keep two arrays, T1 and T2, where all the low-order bytes are in T1 and all the high-order bytes are in T2. That is, what was in T($2k-1$) and T($2k$), respectively, is now in T1(k) and T2(k). There is now no need to multiply k by 2, and, perhaps more importantly, we can have $n \leqslant 256$ rather than $n \leqslant 128$ as in the preceding example. (The reason in both cases, of course, is that our indices into the arrays T, T1, and T2 must, in each case, fit into an eight-bit register, and must therefore be between 0 and 255, inclusive. Arrays with this property may be referred to as *short arrays*.) The code is now

```
LDY  K
LDA  T2-1,Y
STA  IA+1
LDA  T1-1,Y
TAY
LDA  (IA),Y
```

under the same assumptions as before.

In either case, incrementing the calculated indirect address is also speeded up. Incrementing the 16-bit quantity in IA and IA+1, if we needed that, would have been performed by

```
     INC  IA
     BNE  NXT
     INC  IA+1
NXT  (next instruction)
```

In the actual case, the incrementation is performed by

```
     INY
     BNE  NXT2
     INC  IA+1
NXT2 (next instruction)
```

which is again both shorter and faster.

It is true that, in this case, if we know that none of our strings has length greater than 256, we can do even better by using straightforward post-indexing rather than simulated non-indexing. The setup takes a little longer—we have to write something like

```
LDA  K
ASL
TAY
LDA  T-1,Y
```

```
STA  IA2+1
LDA  T-2,Y
STA  IA2
LDY  #0
LDA  (IA2),Y
```

—but now a simple INY suffices for incrementing the indirect address. Note that we carefully set up two further page-zero locations here, IA2 and IA2+1, because IA2, unlike IA, is *not* kept at the constant value of zero.

Another, even more common, application of these ideas is in indexing *long arrays* (this is, arrays of more than 256 bytes). Suppose that U is such an array and we wish to index U(J), where J has just been calculated. Here both U and J are 16-bit quantities. Assuming that #U and /U represent, respectively, the (constant) lower half and upper half of the address of U, we can add U and J and store the result at IA2 by the following code:

```
CLC
LDA  #U
ADC  J
STA  IA2
LDA  /U
ADC  J+1
STA  IA2+1
```

and then, if we had indirect addressing without indexing, we would be ready to apply it. As it is, all we have to write is

```
CLC
LDA  #U
ADC  J
TAY
LDA  /U
ADC  J+1
STA  IA+1
```

with IA set permanently to zero as before, and we are ready to use an instruction like LDA (IA),Y (again, this is both shorter and faster than what preceded it).

If we can set aside another two bytes in page zero for use only with *this particular array* U, we can do even better than that. Let us call them IA3 and IA3+1; the permanent contents of IA3 are the low-order byte of the address of U, rather than zero. This can be set up with a declaration like

```
IA3 ADR U
```

which actually sets up *both* IA3 and IA3+1, although only IA3 will be needed. We can now save two instructions each by writing

```
CLC
LDA  /U
ADC  J+1
STA  IA3+1
LDY  J
```

and now LDA (IA3),Y will load U(J) into the A register. Why does that work? Let us make the following abbreviations:

- p – high-order byte of address of U (i.e., /U)
- q – low-order byte of address of U (i.e., #U)
- r – high-order byte of J (contents of location J+1)
- s – low-order byte of J (contents of location J)

The 16-bit quantities we are adding are $256p+q$ and $256r+s$. The two bytes IA3 and IA3+1 contain, respectively, q and $p+r$, so that this 16-bit quantity is $256(p+r)+q$. To this we add s by post-indexing, so that the computed address is actually $(256(p+r)+q)+s = (256p+q)+(256r+s)$, or J plus the address of U.

The above may be extended immediately to arrays U of two-byte or four-byte quantities. A curious intermittent error may arise, however, if three-byte quantitites are used. We would, of course, multiply the 16-bit quantity in J and J+1 by 3 before using it, and then we would increment Y by one (using INY) to get the second byte set up, and again to get the third byte set up. This works fine, 99% of the time. However, if J is 85, so that 3J is 255, then incrementing Y by one is not enough, since there is carry into the high-order byte of 3J. We must test for nonzero after the increment, and then increment IA3+1 otherwise. (Similar problems arise for J = 170, 341, 426, etc.) It should not be hard to see that there will never be carry of this kind for an array of n-byte quantities where n is 2, 4, 8, or in general any power of 2.

In the above case it was assumed that U(0), rather than U(1), is contained at U; if this is not the case, then the address of U minus 1 must be substituted, in the above, for the address of U. We assume throughout, of course, that all 16-bit quantities are kept with the low-order byte first, as is customary on the 6502. The concepts of serial and parallel array are discussed further in [1]. Finally, users of the LISA assembler for the 6502 should substitute $0, $1, and $2 for 0, 1, and 2 respectively, throughout the coding examples above.

Reference: Maurer, W.D., *Programming*, Holden-Day, 1972, pp. 74-75.

About himself, Prof. Maurer has this to say:

I've worked for CSC, Lockheed, Project MAC, UC Berkeley (as an Assistant Professor) and The George Washington University, Washington, D.C. (as an Associate Professor and now a full Professor). I've written a book called Programming *as well as* The Programmer's Introduction to LISP *and* The Programmer's Introduction to SNOBOL. *In my spare time I write songs (My Pet Rock Is In Love, Dracula's Mood Ring, etc.) and have them performed by a group of weird people called The Hexagon Club.*

LISTING

```
JPRINT ""
!LOAD PPR5
BLOAD  PPR5,A$2000
OX!LIST
1                ORG $18
2                ADR U
3                ORG $0800
4                LDA #L2
5                STA AD
6                LDA /L2
7                STA AD+$1
8                LDA #$0
9                STA P
10               LDY AD
11               STY IA
12               LDY AD+$1
13               STY IA+$1
14               JMP (IA)
15       L2      LDA #P
16               STA AD
17               LDA /P
18               STA AD+$1
19               LDY AD
20               STY IA
21               LDY AD+$1
22               STY IA+$1
23               LDY #$0
24               LDA #$37
25               STA (IA),Y
26               JSR TS1
27               LDA #$0
28               STA IA
29               LDY AD+$1
30               STY IA+$1
31               LDY AD
32               LDA #$37
33               STA (IA),Y
34               JSR TS1
35               LDA #$11
36               STA K
37               LDA #P
38               STA T+$20
39               STA T1+$10
40               LDA /P
41               STA T+$21
42               STA T2+$10
43               LDA K
44               ASL
45               TAY
46               LDA T-$1,Y
47               STA IA+$1
48               LDA T-$2,Y
49               TAY
50               LDA (IA),Y
51               CMP #$0
52               BEQ L4
53               BRK
54       L4      LDA #$15
55               STA P
56               LDY K
57               LDA T2-$1,Y
58               STA IA+$1
59               LDA T1-$1,Y
60               TAY
61               LDA (IA),Y
```

```
                CMP #$15
                BEQ L5
                BRK
L5              LDA #$53
                STA P+$1
                LDX IA
                LDA #P
                STA IA
                LDA /P
                STA IA+$1
                INC IA
                BNE NXT
                INC IA+$1
NXT             LDY #$0
                LDA (IA),Y
                CMP #$53
                BEQ L6
                BRK
L6              STX IA
                LDY #P
                INY
                BNE NXT2
                INC IA+$1
NXT2            LDA (IA),Y
                CMP #$53
                BNE BK
                LDA #$7
                STA P
                LDA #$9
                STA P2
                LDA K
                ASL
                TAY
                LDA T-$1,Y
                STA IA2+$1
                LDA T-$2,Y
                STA IA2
                LDY #$0
                LDA (IA2),Y
                CMP #$7
                BNE BK
                INY
                LDA (IA2),Y
                CMP #$9
                BNE BK
                LDA #$51
                STA J
                LDA #$3
                STA J+$1
                LDA #$14
                STA U+$351
                CLC
                LDA #U
                ADC J
                STA IA2
                LDA /U
                ADC J+$1
                STA IA2+$1
                LDY #$0
                LDA (IA2),Y
                CMP #$14
                BNE BK
                CLC
                LDA #U
                ADC J
                TAY
                LDA /U
                ADC J+$1
                STA IA+$1
                LDA (IA),Y
                CMP #$14
                BNE BK
                CLC
                LDA /U
                ADC J+$1
                STA IA3+$1
                LDY J
                LDA (IA3),Y
                CMP #$14
                BNE BK
EN              BRK
BK              BRK
P               ORG *+$1
P2              ORG *+$1
TS1             LDA #$37
                CMP P
                BNE BK
                LDA #$0
                STA P
                RTS
IA3             EPZ $18
IA              EPZ $1A
IA2             EPZ $1C
AD              ADR L2
K               ORG *+$1
T               ORG *+$50
T1              ORG *+$50
T2              ORG *+$50
U               ORG *+$500
J               ORG *+$2
                END
!ASM
**END OF PASS 1
**END OF PASS 2
0018            1               ORG $18
0018 320A       2               ADR U
0800            3               ORG $0800
0800 A91C       4               LDA #L2
0802 8D3F09     5               STA AD
0805 A908       6               LDA /L2
0807 8D4009     7               STA AD+$1
080A A900       8               LDA #$0
080C 8D3009     9               STA P
080F AC3F09     10              LDY AD
0812 841A       11              STY IA
0814 AC4009     12              LDY AD+$1
0817 841B       13              STY IA+$1
0819 6C1A00     14              JMP (IA)
081C A930       15      L2      LDA #P
081E 8D3F09     16              STA AD
0821 A909       17              LDA /P
0823 8D4009     18              STA AD+$1
0826 AC3F09     19              LDY AD
0829 841A       20              STY IA
082B AC4009     21              LDY AD+$1
082E 841B       22              STY IA+$1
0830 A000       23              LDY #$0
0832 A937       24              LDA #$37
0834 911A       25              STA (IA),Y
0836 203209     26              JSR TS1
0839 A900       27              LDA #$0
083B 851A       28              STA IA
083D AC4009     29              LDY AD+$1
0840 841B       30              STY IA+$1
0842 AC3F09     31              LDY AD
0845 A937       32              LDA #$37
0847 911A       33              STA (IA),Y
0849 203209     34              JSR TS1
084C A911       35              LDA #$11
084E 8D4109     36              STA K
0851 A930       37              LDA #P
0853 8D6209     38              STA T+$20
0856 8DA209     39              STA T1+$10
0859 A909       40              LDA /P
085B 8D6309     41              STA T+$21
085E 8DF209     42              STA T2+$10
0861 AD4109     43              LDA K
0864 0A         44              ASL
0865 A8         45              TAY
0866 B94109     46              LDA T-$1,Y
0869 851B       47              STA IA+$1
086B B94009     48              LDA T-$2,Y
086E A8         49              TAY
086F B11A       50              LDA (IA),Y
0871 C900       51              CMP #$0
0873 F001       52              BEQ L4
0875 00         53              BRK
0876 A915       54      L4      LDA #$15
0878 8D3009     55              STA P
087B AC4109     56              LDY K
087E B9E109     57              LDA T2-$1,Y
0881 851B       58              STA IA+$1
0883 B99109     59              LDA T1-$1,Y
0886 A8         60              TAY
0887 B11A       61              LDA (IA),Y
0889 C915       62              CMP #$15
088B F001       63              BEQ L5
088D 00         64              BRK
088E A953       65      L5      LDA #$53
0890 8D3109     66              STA P+$1
0893 A61A       67              LDX IA
```

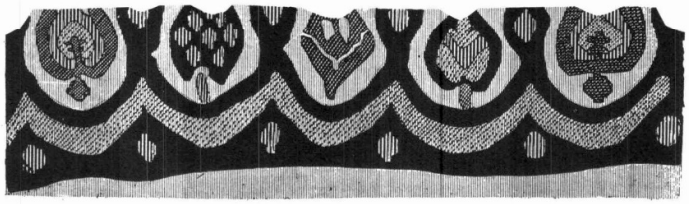
```
0895 A930    68          LDA #P
0897 851A    69          STA IA
0899 A909    70          LDA /P
089B 851B    71          STA IA+$1
089D E61A    72          INC IA
089F D002    73          BNE NXT
08A1 E61B    74          INC IA+$1
08A3 A000    75  NXT     LDY #$0
08A5 B11A    76          LDA (IA),Y
08A7 C953    77          CMP #$53
08A9 F001    78          BEQ L6
08AB 00      79          BRK
08AC 861A    80  L6      STX IA
08AE A030    81          LDY #P
08B0 C8      82          INY
08B1 D002    83          BNE NXT2
08B3 E61B    84          INC IA+$1
08B5 B11A    85  NXT2    LDA (IA),Y
08B7 C953    86          CMP #$53
08B9 D074    87          BNE BK
08BB A907    88          LDA #$7
08BD 8D3009  89          STA P
08C0 A909    90          LDA #$9
08C2 8D3109  91          STA P2
08C5 AD4109  92          LDA K
08C8 0A      93          ASL
08C9 A8      94          TAY
08CA B94109  95          LDA T-$1,Y
08CD 851D    96          STA IA2+$1
08CF B94009  97          LDA T-$2,Y
08D2 851C    98          STA IA2
08D4 A000    99          LDY #$0
08D6 B11C   100          LDA (IA2),Y
08D8 C907   101          CMP #$7
08DA D053   102          BNE BK
08DC C8     103          INY
08DD B11C   104          LDA (IA2),Y
08DF C909   105          CMP #$9
08E1 D04C   106          BNE BK
08E3 A951   107          LDA #$51
08E5 8D320F 108          STA J
08E8 A903   109          LDA #$3
08EA 8D330F 110          STA J+$1
08ED A914   111          LDA #$14
08EF 8D830D 112          STA U+$351
08F2 18     113          CLC
08F3 A932   114          LDA #U
08F5 6D320F 115          ADC J
08F8 851C   116          STA IA2
08FA A90A   117          LDA /U
08FC 6D330F 118          ADC J+$1
08FF 851D   119          STA IA2+$1
0901 A000   120          LDY #$0
0903 B11C   121          LDA (IA2),Y
0905 C914   122          CMP #$14
0907 D026   123          BNE BK
0909 18     124          CLC
090A A932   125          LDA #U
090C 6D320F 126          ADC J
090F A8     127          TAY
0910 A90A   128          LDA /U
0912 6D330F 129          ADC J+$1
0915 851B   130          STA IA+$1
0917 B11A   131          LDA (IA),Y
0919 C914   132          CMP #$14
091B D012   133          BNE BK
091D 18     134          CLC
091E A90A   135          LDA /U
0920 6D330F 136          ADC J+$1
0923 8519   137          STA IA3+$1
0925 AC320F 138          LDY J
0928 B118   139          LDA (IA3),Y
092A C914   140          CMP #$14
092C D001   141          BNE BK
092E 00     142  EN      BRK
092F 00     143  BK      BRK
0931        144  P       ORG *+$1
0932        145  P2      ORG *+$1
0932 A937   146  TS1     LDA #$37
0934 CD3009 147          CMP P
0937 D0F6   148          BNE BK
0939 A900   149          LDA #$0
093B 8D3009 150          STA P
093E 60     151          RTS
093F        152  IA3     EPZ $18
093F        153  IA      EPZ $1A
093F        154  IA2     EPZ $1C
093F 1C08   155  AD      ADR L2
0942        156  K       ORG *+$1
0992        157  T       ORG *+$50
09E2        158  T1      ORG *+$50
0A32        159  T2      ORG *+$50
0F32        160  U       ORG *+$500
0F34        161  J       ORG *+$2
            162          END
***** END OF ASSEMBLY


            ************************
            *                      *
            * SYMBOL TABLE -- V 1 0 *
            *                      *
            ************************
LABEL.  LOC.  LABEL.  LOC.  LABEL.  LOC.
IA3     0018  IA      001A  IA2     001C
L2      081C  L4      0876  L5      088E
NXT     08A3  L6      08AC  NXT2    08B5
EN      092E  BK      092F  P       0930
P2      0931  TS1     0932  AD      093F
K       0941  T       0942  T1      0992
T2      09E2  U       0A32  J       0F32
!
```

```
*0800T
0800-   A9 1C       LDA   #$1C
 A=1C X=00 Y=51 P=31 S=E3
0802-   8D 3F 09    STA   $093F
 A=1C X=00 Y=51 P=31 S=E3
0805-   A9 08       LDA   #$08
 A=08 X=00 Y=51 P=31 S=E3
0807-   8D 40 09    STA   $0940
 A=08 X=00 Y=51 P=31 S=E3
080A-   A9 00       LDA   #$00
 A=00 X=00 Y=51 P=33 S=E3
080C-   8D 30 09    STA   $0930
 A=00 X=00 Y=51 P=33 S=E3
080F-   AC 3F 09    LDY   $093F
 A=00 X=00 Y=1C P=31 S=E3
0812-   84 1A       STY   $1A
 A=00 X=00 Y=1C P=31 S=E3
0814-   AC 40 09    LDY   $0940
 A=00 X=00 Y=08 P=31 S=E3
0817-   84 1B       STY   $1B
 A=00 X=00 Y=08 P=31 S=E3
0819-   6C 1A 00    JMP   ($001A)
 A=00 X=00 Y=08 P=31 S=E3
081C-   A9 30       LDA   #$30
 A=30 X=00 Y=08 P=31 S=E3
081E-   8D 3F 09    STA   $093F
 A=30 X=00 Y=08 P=31 S=E3
0821-   A9 09       LDA   #$09
 A=09 X=00 Y=08 P=31 S=E3
0823-   8D 40 09    STA   $0940
 A=09 X=00 Y=08 P=31 S=E3
0826-   AC 3F 09    LDY   $093F
 A=09 X=00 Y=30 P=31 S=E3
0829-   84 1A       STY   $1A
 A=09 X=00 Y=30 P=31 S=E3
082B-   AC 40 09    LDY   $0940
 A=09 X=00 Y=09 P=31 S=E3
082E-   84 1B       STY   $1B
 A=09 X=00 Y=09 P=31 S=E3
0830-   A0 00       LDY   #$00
 A=09 X=00 Y=00 P=33 S=E3
0832-   A9 37       LDA   #$37
 A=37 X=00 Y=00 P=31 S=E3
0834-   91 1A       STA   ($1A),Y
 A=37 X=00 Y=00 P=31 S=E3
0836-   20 32 09    JSR   $0932
 A=37 X=00 Y=00 P=31 S=E3
0932-   A9 37       LDA   #$37
 A=37 X=00 Y=00 P=31 S=E1
0934-   CD 30 09    CMP   $0930
 A=37 X=00 Y=00 P=33 S=E1
0937-   D0 F6       BNE   $092F
 A=37 X=00 Y=00 P=33 S=E1
0939-   A9 00       LDA   #$00
 A=00 X=00 Y=00 P=33 S=E1
```

```
093B-   8D 30 09     STA   $0930
 A=00 X=00 Y=00 P=33 S=E1
093E-   60           RTS
 A=00 X=00 Y=00 P=33 S=E1
0839-   A9 00        LDA   #$00
 A=00 X=00 Y=00 P=33 S=E3
083B-   85 1A        STA   $1A
 A=00 X=00 Y=00 P=33 S=E3
083D-   AC 40 09     LDY   $0940
 A=00 X=00 Y=09 P=31 S=E3
0840-   84 1B        STY   $1B
 A=00 X=00 Y=09 P=31 S=E3
0842-   AC 3F 09     LDY   $093F
 A=00 X=00 Y=30 P=31 S=E3
0845-   A9 37        LDA   #$37
 A=37 X=00 Y=30 P=31 S=E3
0847-   91 1A        STA   ($1A),Y
 A=37 X=00 Y=30 P=31 S=E3
0849-   20 32 09     JSR   $0932
 A=37 X=00 Y=30 P=31 S=E3
0932-   A9 37        LDA   #$37
 A=37 X=00 Y=30 P=31 S=E1
0934-   CD 30 09     CMP   $0930
 A=37 X=00 Y=30 P=33 S=E1
0937-   D0 F6        BNE   $092F
 A=37 X=00 Y=30 P=33 S=E1
0939-   A9 00        LDA   #$00
 A=00 X=00 Y=30 P=33 S=E1
093B-   8D 30 09     STA   $0930
 A=00 X=00 Y=30 P=33 S=E1
093E-   60           RTS
 A=00 X=00 Y=30 P=33 S=E1
084C-   A9 11        LDA   #$11
 A=11 X=00 Y=30 P=31 S=E3
084E-   8D 41 09     STA   $0941
 A=11 X=00 Y=30 P=31 S=E3
0851-   A9 30        LDA   #$30
 A=30 X=00 Y=30 P=31 S=E3
0853-   8D 62 09     STA   $0962
 A=30 X=00 Y=30 P=31 S=E3
0856-   8D A2 09     STA   $09A2
 A=30 X=00 Y=30 P=31 S=E3
0859-   A9 09        LDA   #$09
 A=09 X=00 Y=30 P=31 S=E3
085B-   8D 63 09     STA   $0963
 A=09 X=00 Y=30 P=31 S=E3
085E-   8D F2 09     STA   $09F2
 A=09 X=00 Y=30 P=31 S=E3
0861-   AD 41 09     LDA   $0941
 A=11 X=00 Y=30 P=31 S=E3
0864-   0A           ASL
 A=22 X=00 Y=30 P=30 S=E3
0865-   A8           TAY
 A=22 X=00 Y=22 P=30 S=E3
0866-   B9 41 09     LDA   $0941,Y
 A=09 X=00 Y=22 P=30 S=E3
0869-   85 1B        STA   $1B
 A=09 X=00 Y=22 P=30 S=E3
086B-   B9 40 09     LDA   $0940,Y
 A=30 X=00 Y=22 P=30 S=E3
086E-   A8           TAY
 A=30 X=00 Y=30 P=30 S=E3
086F-   B1 1A        LDA   ($1A),Y
 A=00 X=00 Y=30 P=32 S=E3
0871-   C9 00        CMP   #$00
 A=00 X=00 Y=30 P=33 S=E3
0873-   F0 01        BEQ   $0876
 A=00 X=00 Y=30 P=33 S=E3
0876-   A9 15        LDA   #$15
 A=15 X=00 Y=30 P=31 S=E3
0878-   8D 30 09     STA   $0930
 A=15 X=00 Y=30 P=31 S=E3
087B-   AC 41 09     LDY   $0941
 A=15 X=00 Y=11 P=31 S=E3
087E-   B9 E1 09     LDA   $09E1,Y
 A=09 X=00 Y=11 P=31 S=E3
0881-   85 1B        STA   $1B
 A=09 X=00 Y=11 P=31 S=E3
0883-   B9 91 09     LDA   $0991,Y
 A=30 X=00 Y=11 P=31 S=E3
0886-   A8           TAY
 A=30 X=00 Y=30 P=31 S=E3
0887-   B1 1A        LDA   ($1A),Y
 A=15 X=00 Y=30 P=31 S=E3
0889-   C9 15        CMP   #$15
 A=15 X=00 Y=30 P=33 S=E3
088B-   F0 01        BEQ   $088E
 A=15 X=00 Y=30 P=33 S=E3
088E-   A9 53        LDA   #$53
 A=53 X=00 Y=30 P=31 S=E3
0890-   8D 31 09     STA   $0931
 A=53 X=00 Y=30 P=31 S=E3
0893-   A6 1A        LDX   $1A
 A=53 X=00 Y=30 P=33 S=E3
0895-   A9 30        LDA   #$30
 A=30 X=00 Y=30 P=31 S=E3
0897-   85 1A        STA   $1A
 A=30 X=00 Y=30 P=31 S=E3
0899-   A9 09        LDA   #$09
 A=09 X=00 Y=30 P=31 S=E3
089B-   85 1B        STA   $1B
 A=09 X=00 Y=30 P=31 S=E3
089D-   E6 1A        INC   $1A
 A=09 X=00 Y=30 P=31 S=E3
089F-   D0 02        BNE   $08A3
 A=09 X=00 Y=30 P=31 S=E3
08A3-   A0 00        LDY   #$00
 A=09 X=00 Y=00 P=33 S=E3
08A5-   B1 1A        LDA   ($1A),Y
 A=53 X=00 Y=00 P=31 S=E3
08A7-   C9 53        CMP   #$53
 A=53 X=00 Y=00 P=33 S=E3
08A9-   F0 01        BEQ   $08AC
 A=53 X=00 Y=00 P=33 S=E3
08AC-   86 1A        STX   $1A
 A=53 X=00 Y=00 P=33 S=E3
08AE-   A0 30        LDY   #$30
 A=53 X=00 Y=30 P=31 S=E3
08B0-   C8           INY
 A=53 X=00 Y=31 P=31 S=E3
08B1-   D0 02        BNE   $08B5
 A=53 X=00 Y=31 P=31 S=E3
08B5-   B1 1A        LDA   ($1A),Y
 A=53 X=00 Y=31 P=31 S=E3
08B7-   C9 53        CMP   #$53
 A=53 X=00 Y=31 P=33 S=E3
08B9-   D0 74        BNE   $092F
 A=53 X=00 Y=31 P=33 S=E3
08BB-   A9 07        LDA   #$07
 A=07 X=00 Y=31 P=31 S=E3
08BD-   8D 30 09     STA   $0930
 A=07 X=00 Y=31 P=31 S=E3
08C0-   A9 09        LDA   #$09
 A=09 X=00 Y=31 P=31 S=E3
08C2-   8D 31 09     STA   $0931
 A=09 X=00 Y=31 P=31 S=E3
08C5-   AD 41 09     LDA   $0941
 A=11 X=00 Y=31 P=31 S=E3
08C8-   0A           ASL
 A=22 X=00 Y=31 P=30 S=E3
08C9-   A8           TAY
 A=22 X=00 Y=22 P=30 S=E3
08CA-   B9 41 09     LDA   $0941,Y
 A=09 X=00 Y=22 P=30 S=E3
08CD-   85 1D        STA   $1D
 A=09 X=00 Y=22 P=30 S=E3
08CF-   B9 40 09     LDA   $0940,Y
 A=30 X=00 Y=22 P=30 S=E3
08D2-   85 1C        STA   $1C
 A=30 X=00 Y=22 P=30 S=E3
08D4-   A0 00        LDY   #$00
 A=30 X=00 Y=00 P=32 S=E3
08D6-   B1 1C        LDA   ($1C),Y
 A=07 X=00 Y=00 P=30 S=E3
08D8-   C9 07        CMP   #$07
 A=07 X=00 Y=00 P=33 S=E3
08DA-   D0 53        BNE   $092F
 A=07 X=00 Y=00 P=33 S=E3
08DC-   C8           INY
 A=07 X=00 Y=01 P=31 S=E3
08DD-   B1 1C        LDA   ($1C),Y
 A=09 X=00 Y=01 P=31 S=E3
08DF-   C9 09        CMP   #$09
 A=09 X=00 Y=01 P=33 S=E3
08E1-   D0 4C        BNE   $092F
 A=09 X=00 Y=01 P=33 S=E3
08E3-   A9 51        LDA   #$51
 A=51 X=00 Y=01 P=31 S=E3
```

```
08E5-   8D 32 0F    STA   $0F32
 A=51 X=00 Y=01 P=31 S=E3
08E8-   A9 03       LDA   #$03
 A=03 X=00 Y=01 P=31 S=E3
08EA-   8D 33 0F    STA   $0F33
 A=03 X=00 Y=01 P=31 S=E3
08ED-   A9 14       LDA   #$14
 A=14 X=00 Y=01 P=31 S=E3
08EF-   8D 83 0D    STA   $0D83
 A=14 X=00 Y=01 P=31 S=E3
08F2-   18          CLC
 A=14 X=00 Y=01 P=30 S=E3
08F3-   A9 32       LDA   #$32
 A=32 X=00 Y=01 P=30 S=E3
08F5-   6D 32 0F    ADC   $0F32
 A=83 X=00 Y=01 P=F0 S=E3
08F8-   85 1C       STA   $1C
 A=83 X=00 Y=01 P=F0 S=E3
08FA-   A9 0A       LDA   #$0A
 A=0A X=00 Y=01 P=70 S=E3
08FC-   6D 33 0F    ADC   $0F33
 A=0D X=00 Y=01 P=30 S=E3
08FF-   85 1D       STA   $1D
 A=0D X=00 Y=01 P=30 S=E3
0901-   A0 00       LDY   #$00
 A=0D X=00 Y=00 P=32 S=E3
0903-   B1 1C       LDA   ($1C),Y
 A=14 X=00 Y=00 P=30 S=E3
0905-   C9 14       CMP   #$14
 A=14 X=00 Y=00 P=33 S=E3
0907-   D0 26       BNE   $092F
 A=14 X=00 Y=00 P=33 S=E3
0909-   18          CLC
 A=14 X=00 Y=00 P=32 S=E3
090A-   A9 32       LDA   #$32
 A=32 X=00 Y=00 P=30 S=E3
090C-   6D 32 0F    ADC   $0F32
 A=83 X=00 Y=00 P=F0 S=E3
090F-   A8          TAY
 A=83 X=00 Y=83 P=F0 S=E3
0910-   A9 0A       LDA   #$0A
 A=0A X=00 Y=83 P=70 S=E3
0912-   6D 33 0F    ADC   $0F33
 A=0D X=00 Y=83 P=30 S=E3
0915-   85 1B       STA   $1B
 A=0D X=00 Y=83 P=30 S=E3
0917-   B1 1A       LDA   ($1A),Y
 A=14 X=00 Y=83 P=30 S=E3
0919-   C9 14       CMP   #$14
 A=14 X=00 Y=83 P=33 S=E3
091B-   D0 12       BNE   $092F
 A=14 X=00 Y=83 P=33 S=E3
091D-   18          CLC
 A=14 X=00 Y=83 P=32 S=E3
091E-   A9 0A       LDA   #$0A
 A=0A X=00 Y=83 P=30 S=E3
0920-   6D 33 0F    ADC   $0F33
 A=0D X=00 Y=83 P=30 S=E3
0923-   85 19       STA   $19
 A=0D X=00 Y=83 P=30 S=E3
0925-   AC 32 0F    LDY   $0F32
 A=0D X=00 Y=51 P=30 S=E3
0928-   B1 18       LDA   ($18),Y
 A=14 X=00 Y=51 P=30 S=E3
092A-   C9 14       CMP   #$14
 A=14 X=00 Y=51 P=33 S=E3
092C-   D0 01       BNE   $092F
 A=14 X=00 Y=51 P=33 S=E3
092E-   00          BRK
092E-    A=14 X=00 Y=51 P=33 S=E3
*0800G
0930-    A=14 X=00 Y=51 P=33 S=DF
```



This interface will allow the USART, package 3A, and the software which controls it to interface the H-14 at any baud rate up to and including 4800.

Sincerely,
Robert Rennard

2281 Cobble Stone Court
Dayton, OH 45431

CP/M COMPATIBLE SOFTWARE MARKETING

Dear *DDJ*:

As the developers of CP/M and MP/M, Digital Research is preparing a list of vendors of CP/M-compatible software. We would appreciate the help of readers of your magazine in compiling this list for distribution to all interested persons who contact us.

If any readers are currently marketing CP/M-compatible software, please send us any or all literature pertaining to your software which controls it to interface the H-14 at any baud rate up to and including 4800.

Thank you,
Marilyn Darling

Digital Research
P.O. Box 579
Pacific Grove, CA 93950

AT ODDS WITH *DDJ*

Dear *DDJ*:

I have received my copies of Volume 5, Issue 8 of *DDJ*, which contains my article (*A Note On 6502 Indirect Addressing*), and I am slightly upset.

I have about 70 papers to my credit, in all kinds of journals, and this is the *very first time* that I have not had the courtesy of having galley proofs to review before an article of mine is printed.

I understand about your policy of quick publication; I want to see my ideas on programming published as quickly as possible, too—but not with typists' mistakes still in them! Typists are not infallible, nor should they be expected to be. For the record, the mistakes were as follows:

P. 26 col. 1 line 37: "is keep" should be "is to keep"
P. 26 col. 3 line 34: left paren should line up under I
P. 27 col. 1 line 57: "each by" should be "each time by"
P. 27 col. 3 line 11: [1] refers to *reference number 1*, which is not marked as such. (I don't mind if you want to set up your own conventional notation for references to journal papers and books, but it has to be consistent with itself.)

None of these turned out to be really serious, in this case—which is why I am slightly, rather than considerably, upset.

Sincerely,
Professor W. D. Maurer

George Washington University
S.E.A.S.
Washington, DC 20052

Prof. Maurer's objection is well taken and the subject of this month's editorial (see p. 1). —*JP*

SIGNED COMPARISON: ALTERNATE ROUTINE

Dear *DDJ*:

While scanning the listing of the runtime library for the Small C compiler in issue #48, I happened to notice the signed compare routine ("cccmp"). This routine performs a comparison of two 16-bit signed numbers (two's complement) by subtracting one of the numbers from the other and then examining the sign of the result. This technique works fine as long as the difference between the numbers being compared is less than or equal to 32767; however, if the numbers differ by more than 32767, the result of the compare will be incorrect due to arithmetic overflow during the calculation.

A different way to accomplish a signed comparison is to invert the sign bits of the two numbers being compared and then to perform an unsigned compare using the new numbers. A routine using this technique is shown below:

```
; Common routine to perform a signed compare of DE & HL.
; This routine compares DE & HL and sets the conditions:
;    Carry reflects sign of difference (set if DE < HL)
;    Zero/non-zero set according to equality.
;
cccmp:  MOV   A,H      ; invert sign of HL.
        XRI   80H
        MOV   H,A

        MOV   A,D      ; invert sign of DE.
        XRI   80H

        CMP   H        ; compare msbs.
        JNZ   cccmpl   ; unequal—comparison done.

        MOV   A,E      ; equal—compare lsbs.
        CMP   L

cccmpl: LXI   H,1      ; (required by Small C).
        RET
```

This new method works by mapping the signed number domain into the unsigned number domain, where 16-bit comparisons may be easily made, before making the comparison. A few examples of number mappings are shown below:

Decimal number	Hex number before map	Hex number after map
-32768	8000	0000
-32767	8001	0001
.		
-1	FFFF	7FFF
0	0000	8000
1	0001	8001
.		
32767	7FFF	FFFF

Now, does this solve a problem or just create a new problem? Consider the two expressions shown below, which appear to be equivalent:

Expression	New result	Old result	"Correct" result
(-4000 < 30000)	True	False	True
(-4000 - 30000 < 0)	False	False	False

Because of arithmetic overflow in the second expression, where "-4000 - 30000" is correctly interpreted as 31536 (not -34000), the expressions have different values. The new "improved" compare yields the differing results, depending upon how the expression is formed; whereas the old compare yields identical (but incorrect in one case) results no matter which way the expression is formed. Which attibute appeals to you more: correctness or apparent consistency? I vote for correctness in this case.

Yours truly,
Harry B. Stewart

Neoteric
15816 San Benito Way
Los Gatos, CA 95030