Being able to see what a processor does as it runs a machine code program is a great aid in understanding the program, in fault finding, in testing, and in fact in everything a programmer does when developing some new software. The program given here makes it possible to do this automatically. At each step the contents of the CPU registers, the stack and its pointer are displayed for the corresponding instruction.

J. Ruppert

6502 tracer

program analysis software for Junior Computer and other 6502-based systems

This program is aimed not only at users of the Junior Computer but also at the owners of any 6502-based system. It occupies about ½ K of memory and uses two bytes in page zero. Very few changes are needed to adapt it to a system other than the Junior.

How is it used?

The program operates as a sort of 'step by step monitor'. This means in effect that any program the user wishes to analyse, or debug, is executed instruction by instruction with the contents of registers A, X, and Y, the status register flags (NV DIZC) and the stack pointer being displayed each time. It is notable from the list of flags (NV DIZC) that the 'break' flag is not included; the reason is that the '6502 TRACER' program accepts all instructions except those which are the result of, or which result in, an interrupt (BRK, IRQ and NMI).

As table 3 shows, it is much easier to analyse a program (the example here contains a lot of register and flag manipulations) with the aid of the information displayed by the tracer program in the three right hand columns. The first, at the extreme right, refers to the stack: $FF is the least significant byte of the pointer (the most significant byte is $01). Near the end of the listing there are a few addresses stacked during JSR or RTS instructions. The next column gives the logic levels of the status register flags NV DIZC. Finally, beside this the contents of the A, X, Y and processor registers are to be found. The step by step tracing of the program in these columns is followed in the first two columns by the disassembled listing of the addresses and instructions. The fact that all jumps and branches are included explains why the program returns from address $020D (D0/FA) to address $0209 but the Z flag remains low.

Table 1. 6502 TRACER is an analysis program that must run in RAM, but there is nothing to stop you from storing it in some other kind of memory and simply transferring it to RAM to run it.

table 1

```
JUNIOR

M
HEXDUMP: 500,721
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0500: 58 20 95 06 A9 00 A0 0F 99 13 07 88 D0 FA B9 CC
0510: 06 20 A5 06 C8 C0 36 D0 F5 A9 26 8D 7E 1A A9 05
0520: 8D 7F 1A 4C A2 05 8D 1B 07 68 8D 20 07 68 68 8C
0530: 1C 07 8E 1D 07 BA 8E 14 07 D8 58 A0 03 B9 15 07
0540: 20 A0 06 20 A3 06 C8 C0 06 B0 11 AD 16 07 D0 09
0550: 20 A3 06 20 A3 06 4C 43 05 CE 16 07 C0 09 D0 DD
0560: AD 20 07 29 CF 8D 13 07 A2 08 0E 13 07 90 04 A9
0570: 31 D0 02 A9 2E 20 A5 06 CA D0 EF 20 A3 06 AD 14
0580: 07 20 A0 06 A9 2D 20 A5 06 BA E0 FF B0 14 68 8D
0590: 16 07 20 A0 06 E0 FE B0 05 68 48 20 A0 06 AD 16
05A0: 07 48 A0 00 20 95 06 A5 EE 20 A0 06 A5 ED 20 A0
05B0: 06 20 A3 06 B1 ED 8C 1A 06 8C 1B 06 8C 1A 07 8C
05C0: 19 07 20 A8 06 8C 1E 07 98 8D 16 07 CE 16 07 88
05D0: B1 ED 99 19 06 99 18 07 98 D0 F4 E6 ED D0 02 E6
05E0: EE CE 1E 07 D0 F5 AD 18 07 29 0F D0 13 AD 18 07
05F0: C9 20 F0 29 C9 40 F0 2E C9 60 F0 2E 29 10 D0 62
0600: AD 18 07 C9 4C F0 2C C9 6C F0 3D AE 1D 07 AC 1C
0610: 07 AD 20 07 48 AD 1B 07 28 D0 00 00 00 A5 ED 48
0620: A5 EE 48 4C 33 06 68 8D 20 07 68 85 EE 68 85 ED
0630: 4C 3D 06 AD 1A 06 85 ED AD 1B 06 85 EE A9 00 8D
0640: 19 06 20 9A 06 4C 0B 06 AD 1A 06 85 ED AD 1B 06
0650: 85 EE A0 00 B1 ED AA C8 B1 ED 85 EE 8A 85 ED 4C
0660: 3D 06 AD 20 07 48 AD 18 07 8D 6D 06 28 D0 03 4C
0670: 82 06 58 D8 AD 1A 06 30 11 18 65 ED 85 ED 90 02
0680: E6 EE A9 00 8D 1A 06 4C 00 06 18 65 ED 85 ED B0
0690: F1 C6 EE 90 ED A9 0D 20 A5 06 A9 0A 20 A5 06 60
06A0: 4C 8F 12 A9 20 4C 34 13 A0 01 C9 00 F0 1A C9 40
06B0: F0 16 C9 60 F0 12 A0 03 C9 20 F0 0C 29 1F C9 19
06C0: F0 06 29 0F AA BC 03 07 8C 21 07 60 36 35 30 32
06D0: 20 2D 20 54 52 41 43 45 52 0D 0A 41 44 52 2E 20
06E0: 2D 49 4E 53 54 52 2E 2D 20 3A 41 20 3A 59 20 3A
06F0: 58 20 4E 56 31 31 44 49 5A 43 20 53 54 41 43 4B
0700: 20 0D 0A 02 02 02 01 02 02 02 01 01 02 01 01 03
0710: 03 03 03 80 FB 00 00 00 D0 FD 00 04 71 08 00 00
0720: 31 02
```

How does it work?

The length of this article does not give us the scope to provide a complete source listing of this tracer program, so we will have to be content with the hex dump shown in table 1. It is, however, quite important to have some pointers about how to use the software.

Before a run the start address of the program to be tested must be stored at addresses $00ED and $00EE which act as a pseudo program counter. The program under test may be in back-up memory but the tracer program must be in RAM: as shown here it starts at address $0500. Between addresses $0500 and $0523

several buffer bytes acting as a pseudo stack that starts at $0713 (we will return to this later) are initialized, the column headings are displayed and the IRQ vector is positioned (the IRQ routine begins at address $0526).

The tracing proper starts at $05A2, by displaying the program counter address, loading the op-code, filling the op-field with 00s, and calculating the length of the instruction (the routine used begins at $06A8 and is quite similar to the LENACC routine in the Junior Computer). The op-field is a four-byte zone ($0619...$061C) where the analysis program places in turn each of the instructions of the program under test in order to execute them. As these instructions never contain more than three bytes they are always followed by at least one 00 and this functions as a BRK. Immediately after executing an instruction of the program under test, therefore, this BRK causes the IRQ routine at $0526 to be run.

The pseudo program counter ($00ED and $00EE) is incremented at $05DB. This incrementation depends on the format of the preceding instruction, with the number of bytes making up the instruction being stored in address $071E. Any jump instructions in the program must be filtered out to be dealt with separately and this begins at $05E6. From $060B onwards stacking of registers A, X and Y for the program under test starts. The op-field, located at $0619, contains the instruction to be analysed and because every instruction is always followed by at least one BRK it is also followed immediately by the IRQ routine. As could be expected, this begins by storing the conditions of the processor registers. Then it displays their contents and proceeds to the next instruction.

The special instructions for executing jump commands are located at $061D. The addresses for relative jumps are calculated at $0672 and $068A. The addresses of the Junior Computer's PRBYT and PRCHA routines are contained in $06A1, $06A2, $06A6 and $06A7, so these must be changed if the program is to be used with a different 6502 system.

The commands for printing the headings of the columns are at $06CC to $0702. The format of each instruction that is to be run is determined by comparing it to the values contained in the look-up table located from $0703 to $0712. There are a number of buffers between $0713 and $0721 that are used by the tracer program to store the stack pointer, the contents of the top of the stack, the op-code under test, the number of bytes in the instruction, and so on...

These were the most important points about this program and the rest is easily deciphered with the aid of a disassembler.

table 2

```
JUNIOR

M
HEXDUMP: 200,23A
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
0200: A9 03 A8 AA A9 09 85 00 F8 18 65 00 CA D0 FA 2A
0210: 6A 38 E5 00 88 D0 FA E5 00 D8 F0 00 F0 06 F0 02
0220: F0 04 F0 FC F0 F8 20 30 02 38 EA 4C 35 02 EA EA
0230: 20 34 02 60 60 4C 00 03 4C 00 02

JUNIOR

M
HEXDUMP: 2F0,30F
      0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F
02F0: 00 00 00 00 00 00 00 00 00 00 00 00 B0 06 B0 02
0300: B0 FC B0 F8 6C 07 03 00 02 00 00 00 00 00 00 00
0310:
```

Table 2. These few instructions could be used to test the program of table 1. The result obtained should be the same as table 3.

table 3

```
ED
00ED 27 00.
00EE 09 02.
00EF 1C 500
0500 58 R
6502 - TRACER
ADR. -INSTR.- :A :Y :X NV1BDIZC STACK
0200 A9 03     03 00 00 ........ FF-
0202 A8        03 03 00 ........ FF-
0203 AA        03 03 03 ........ FF-
0204 A9 09     09 03 03 ........ FF-
0206 85 00     09 03 03 ........ FF-
0208 F8        09 03 03 ....1... FF-
0209 18        09 03 03 ....1... FF-
020A 65 00     18 03 03 ....1... FF-
020C CA        18 03 02 ....1... FF-
020D D0 FA     18 03 02 ....1... FF-
0209 18        18 03 02 ....1... FF-
020A 65 00     27 03 02 ....1... FF-
020C CA        27 03 01 ....1... FF-
020D D0 FA     27 03 01 ....1... FF-
0209 18        27 03 01 ....1... FF-
020A 65 00     36 03 01 ....1... FF-
020C CA        36 03 00 ....1.1. FF-
020D D0 FA     36 03 00 ....1.1. FF-
020F 2A        6C 03 00 ....1... FF-
0210 6A        36 03 00 ....1... FF-
0211 38        36 03 00 ....1..1 FF-
0212 E5 00     27 03 00 ....1..1 FF-
0214 88        27 02 00 ....1..1 FF-
0215 D0 FA     27 02 00 ....1..1 FF-
0211 38        27 02 00 ....1..1 FF-
0212 E5 00     18 02 00 ....1..1 FF-
0214 88        18 01 00 ....1..1 FF-
0215 D0 FA     18 01 00 ....1..1 FF-
0211 38        18 01 00 ....1..1 FF-
0212 E5 00     09 01 00 ....1..1 FF-
0214 88        09 00 00 ....1.11 FF-
0215 D0 FA     09 00 00 ....1.11 FF-
0217 E5 00     00 00 00 ....1.11 FF-
0219 D8        00 00 00 ......11 FF-
021A F0 00     00 00 00 ......11 FF-
021C F0 06     00 00 00 ......11 FF-
0224 F0 F8     00 00 00 ......11 FF-
021E F0 02     00 00 00 ......11 FF-
0222 F0 FC     00 00 00 ......11 FF-
0220 F0 04     00 00 00 ......11 FF-
0226
     20 30 02  00 00 00 ......11 FD-0229
0230
     20 34 02  00 00 00 ......11 FB-0233
0234
     60        00 00 00 ......11 FD-0229
0233
     60        00 00 00 ......11 FF-
0229 38        00 00 00 ......11 FF-
022A EA        00 00 00 ......11 FF-
022B
     4C 35 02  00 00 00 ......11 FF-
0235
     4C 00 03  00 00 00 ......11 FF-
0300 B0 FC     00 00 00 ......11 FF-
02FE B0 02     00 00 00 ......11 FF-
0302 B0 F8     00 00 00 ......11 FF-
02FC B0 06     00 00 00 ......11 FF-
0304
     6C 07 03  00 00 00 ......11 FF-
0200 A9 03     03 00 00 .......1 FF-
0202 A8        03 03 00 .......1 FF-
0203 AA
JUNIOR
```

Table 3. This is what should appear on the screen (or printer) if the program of table 2 is run with the aid of TRACER. Before starting the latter at $0500 the start address of the program under test ($0200) must be placed in page zero ($00ED and $00EE).