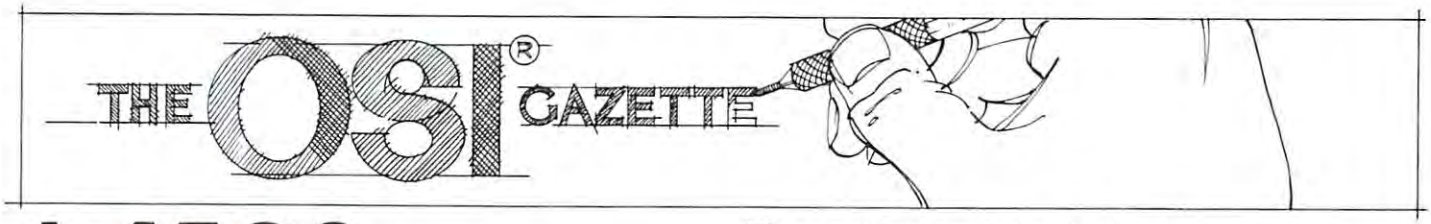


# A 6502 Disassembler

Thomas G. Gordon

As the proud owner of an OSI Superboard II, I was immediately curious to see what made it tick. A little peeking into the Basic-in-Roms via the monitor was enough to convince me that I needed a good disassembler if I was going to get anywhere. Listing 1 is the resultant 6502 disassembler. Although written in Microsoft Basic to run on my OSI, it should run on any 6502 based system with minor modifications (8K memory needed).

Although I feel the program is fairly straight-forward, a few words may be in order to explain its operation.

**Lines 5 - 30** print a greeting and ask for a starting address at which to begin disassembling.

The subroutines at 900-990 and 1000-1095 are hex to decimal and decimal to hex conversion routines, respecitvely.

the subroutine at lines 250-340 inserts the mnemonic op codes into the array R$, dimensioned by line 5. Each mnemonic contains a fourth letter which I call a tag code. The purpose of the tag code is to identify the addressing mode associated with that particular op code. For example, the tag R indicates relative more addressing.

**Lines 40-75** fetch the numerical op code, print the current address in hex, determine if it is a legal op code, (if not, the operator is requested to enter another starting address) and print the hex op code along with its three letter mnemonic.

**Lines 85-150** determine the addressing mode by examining the tag code, and jump to the proper routine to print any associated arguments (data or address) following the op code. These routines are located at lines 600-795.

The disassembler will continue to run until killed by the operator or an invalid op code is found.

```
LISTING 1
5 DIMR$(257)
10 PRINT"6502 DISASSEMBLER":PRINT:PRINT
15 GOSUB250
20 PRINT"ENTER START ADDRESS"
25 PRINT"IN HEX,USE 4 DIGITS."
30 INPUTA$
35 GOSUB900
40 Z=PEEK(S):A=S
55 GOSUB1000:REM-GET HX ADR
60 PRINTER$;TH$;TW$;DE$;" ";
65 IFR$(Z)=""THENPRINT"INVALID CODE":GOTO20
70 A=Z:GOSUB1000
75 PRINTTW$;DE$;" ";LEFT$(R$(Z),3);" ";
85 U$=RIGHT$(R$(Z),1)
90 IFU$=" "THENPRINTU$:S=S+1:GOTO40
95 IFU$="N"THENPRINT"A#";:GOTO600
100 IFU$="A"THENPRINTU$:S=S+1:GOTO40
105 IFU$="Z"THENPRINT"A#";:GOTO625
110 IFU$="#"THENPRINT"#$";:GOTO625
115 IFU$="X"THENPRINT"A#";:GOTO645
120 IFU$="Y"THENPRINT"A#";:GOTO665
125 IFU$="B"THENPRINT" (";:GOTO685
130 IFU$="C"THENPRINT" (";:GOTO700
135 IFU$="U"THENPRINT"A#";:GOTO715
140 IFU$="R"THENPRINT"TO ";:GOTO765
145 IFU$="J"THENPRINT" (";:GOTO735
150 IFU$="V"THENPRINT"A#";:GOTO755
250 FORX=0TO255:READM$:R$(X)=M$:NEXT
255 DATABRK ,ORAB,,,ORAZ,ASLZ,,PHP ,ORA#,ASLA,,,ORAN
260 DATABPLR,ORAC,,,,ORAU,ASLU,,CLC ,ORAY,,,,ORAX
265 DATAJSRN,ANDB,,,BITZ,ANDZ,ROLZ,,PLP ,AND#,ROLA,,BITN
270 DATAANDU,ROLN,,BMIR,ANDC,,,,ANDU,ROLU,,SEC ,ANDY,,,
275 DATAANDX,ROLX,,RTI ,EORB,,,,EORZ,LSRZ,,PHA ,EOR#,LSRA,,JMPN
280 DATAEORN,LSRN,,BVCR,EORC,,,,EORU,LSRU,,CLI ,EORY,,,
285 DATAEORX,LSRX,,RTS ,ADCB,,,,ADCZ,RORZ,,PLA ,ADC#,RORA,,JMPJ
290 DATAADCN,RORN,,BVSR,ADCC,,,,ADCU,RORU,,SEI ,ADCY,,,,ADCX,,,
295 DATASTAB,,,STYZ,STAZ,STXZ,,DEY ,,TXA ,,STYN,STAN,STXN,,BCCR
300 DATASTAC,,,STYU,STAU,STXV,,TYA ,STAY,TXS ,,,STAX,,,LDY#
305 DATALDAB,LDX#,,LDYZ,LDAZ,LDXZ,,TAY ,LDA#,TAX ,,LDYN,LDAN,LDXN
310 DATA,BCSR,LDAC,,,LDYU,LDAU,LDXV,,CLV ,LDAY,TSX ,,LDYX,LDAX
315 DATALDXY,,CPY#,CMPB,,,CPYZ,CMPZ,DECZ,,INY ,CMP#,DEX ,
320 DATACPYN,CMPN,DECN,,BNER,CMPC,,,,CMPU,DECU,,CLD ,CMPY,,,
325 DATACMPX,DECX,,CPX#,SBCB,,,CPXZ,SBCZ,INCZ,,INX ,SBC#,NOP ,
330 DATACPXN,SBCN,INCN,,BEQR,SBCC,,,
335 DATASBCU,INCU,,SED ,SBCY,,,,SBCX,INCX,
340 RETURN
600 A=PEEK(S+2):GOSUB1000
605 PRINTTW$;DE$;
610 A=PEEK(S+1):GOSUB1000
615 PRINTTW$;DE$
620 S=S+3:GOTO40
625 A=PEEK(S+1):GOSUB1000
630 PRINTTW$;DE$
632 S=S+2:GOTO40
645 A=PEEK(S+2):GOSUB1000
650 PRINTTW$;DE$;
655 A=PEEK(S+1):GOSUB1000
660 PRINTTW$;DE$;",X":S=S+3:GOTO40
665 A=PEEK(S+2):GOSUB1000
670 PRINTTW$;DE$;
675 A=PEEK(S+1):GOSUB1000
680 PRINTTW$;DE$;",Y":S=S+3:GOTO40
```

The format of the resultant printout pretty much follows standard assembler notation, with one exception. When relative addressing mode is encountered, the program prints the hex address to which the branch occurs, rather than the hex offset. I found this to be much more convenient when disassembling. Since it is a one pass only disassembler, the use of labels was out, but this works just as well in my opinion.

Finally, listing 2 shows the resultant printout of some of the OSI code beginning at hex FD00, which is the start of the keyboard monitor routine.

```
685 A=PEEK(S+1):GOSUB1000
690 PRINTTW$;DE$;",X)":GOTO632
700 A=PEEK(S+1):GOSUB1000
705 PRINTTW$;DE$;"),Y":GOTO632
715 A=PEEK(S+1):GOSUB1000
720 PRINTTW$;DE$;",X":GOTO632
735 A=PEEK(S+2):GOSUB1000
740 PRINTTW$;DE$;
745 A=PEEK(S+1):GOSUB1000
750 PRINTTW$;DE$;")":S=S+3:GOTO40
755 A=PEEK(S+1):GOSUB1000
760 PRINTTW$;DE$;"Y":GOTO632
765 A=PEEK(S+1):IFA<128THEN790
770 A=255-A
775 A=S+1-A:GOSUB1000
780 PRINTFR$;TH$;TW$;DE$:GOTO632
790 A=S+A+2:GOSUB1000
795 GOTO780
900 B$=LEFT$(A$,1):C$=MID$(A$,2,1):D$=MID$(A$,3,1)
915 E$=MID$(A$,4,1):F$=B$
925 FORX=1TO4
930 IFF$="A"THENA=10:GOTO965
935 IFF$="B"THENA=11:GOTO965
940 IFF$="C"THENA=12:GOTO965
945 IFF$="D"THENA=13:GOTO965
950 IFF$="E"THENA=14:GOTO965
955 IFF$="F"THENA=15:GOTO965
960 A=VAL(F$)
965 IFX=1THENS=S+A*4096:F$=C$
970 IFX=2THENS=S+A*256:F$=D$
975 IFX=3THENS=S+A*16:F$=E$
980 IFX=4THENS=S+A
985 NEXTX
990 RETURN
1000 F=INT(A/4096):REM-D TO H CONVERT
1005 R=A-F*4096
1010 TH=INT(R/256)
1015 R=R-TH*256
1020 TW=INT(R/16)
1025 DE=R-TW*16:H=F
1030 FORX=1TO4
1035 IFH=10THENF$="A":GOTO1070
1040 IFH=11THENF$="B":GOTO1070
1045 IFH=12THENF$="C":GOTO1070
1050 IFH=13THENF$="D":GOTO1070
1055 IFH=14THENF$="E":GOTO1070
1060 IFH=15THENF$="F":GOTO1070
1065 F$=STR$(H)
1070 IFX=1THENFR$=F$:H=TH
1075 IFX=2THENTH$=F$:H=TW
1080 IFX=3THENTW$=F$:H=DE
1085 IFX=4THENDE$=F$
1090 NEXTX
1095 RETURN
```

A last minute correction from the author: Line 1065 works as is for APPLE; for OSI it should read 1065 F$ = MID$(STR$(H),2)

```
LISTING 2
RUN
6502 DISASSEMBLER
ENTER START ADDRESS
IN HEX,USE 4 DIGITS.
? FD00
FD00 8A TXA
FD01 48 PHA
FD02 98 TYA
FD03 48 PHA
FD04 A9 LDA #$01
FD06 20 JSR A=FCBE
FD09 20 JSR A=FCC6
FD0C D0 BNE TO FD13
FD0E 0A ASL A
FD0F D0 BNE TO FD06
FD11 F0 BEQ TO FD66
FD13 4A LSR A
FD14 90 BCC TO FD1F
FD16 2A ROL A
FD17 E0 CPX #$21
FD19 D0 BNE TO FD0E
FD1B A9 LDA #$1B
FD1D D0 BNE TO FD40
FD1F 20 JSR A=FDC8
FD22 98 TYA
FD23 8D STA A=0213
FD26 0A ASL A
FD27 0A ASL A
```