60p

Win a NewBrain

YOUR COMPUTER

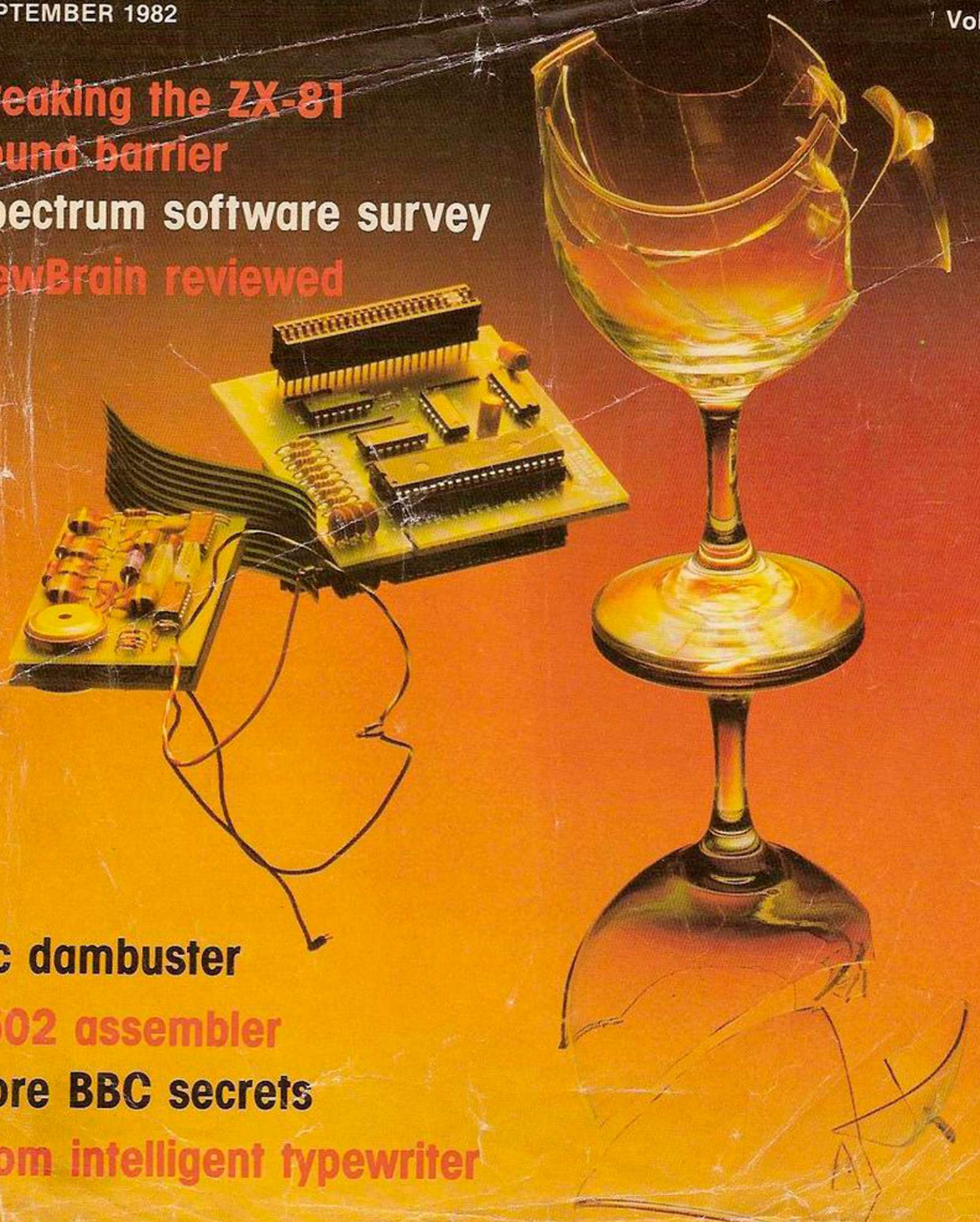
SEPTEMBER 1982

Vol.2 No.9

Breaking the ZX-81 sound barrier

Spectrum software survey

NewBrain reviewed



Vic dambuster

6502 assembler

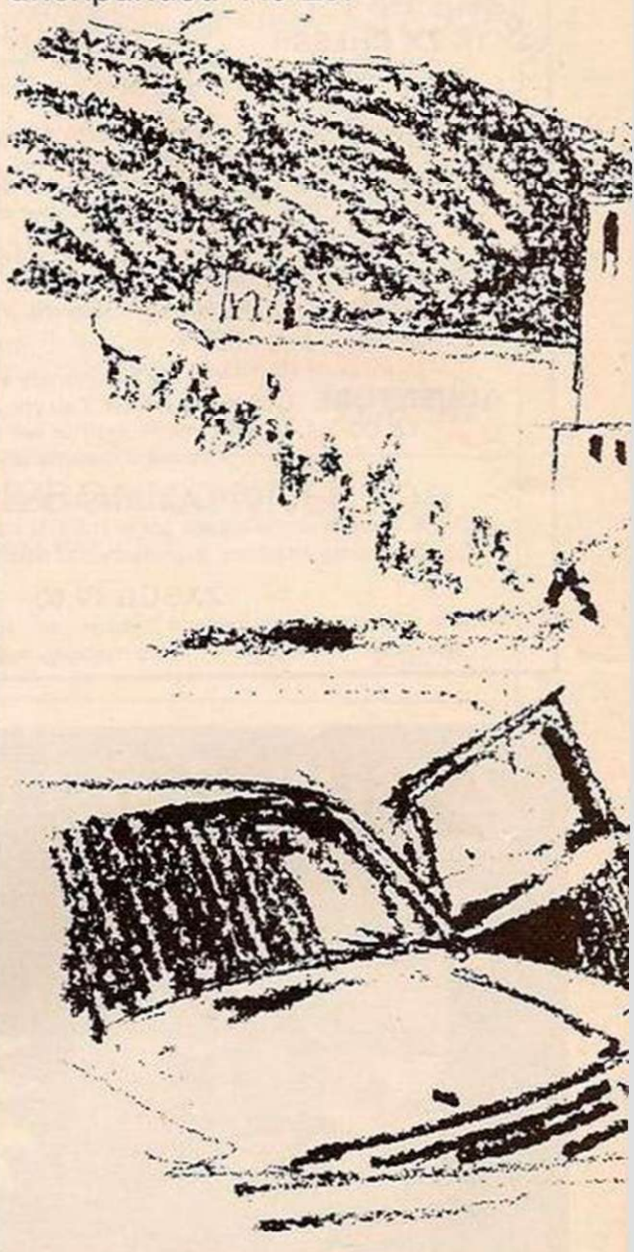
More BBC secrets

Atom intelligent typewriter

GAMES

VIC DA

Can you stop the Nibblers destroying the dam? Dave Shambrook's game is for the unexpanded Vic-20.



THE DAM AT the head of the valley is under attack from an unidentified source. If the dam bursts, the water will escape and flood the valley, killing thousands. Your mission is to destroy the aggressors, code-named Nibblers, and save the dam.

The Nibbler appears on the right-hand side of the screen and moves across towards the dam on the left. To stop the Nibbler and launch your ship at the same time, you hit the space bar. When you are directly above the Nibbler, press the space bar again to drop your bomb.

If you hit the Nibbler, your score increases by one point and the dam has been saved for a little longer. But if you miss, you forfeit a point and the Nibbler lives on to destroy part of the dam. Another Nibbler will then appear on the right-hand side of the screen.

Once the dam has been totally breached, the water will escape and flood the valley, and you have failed in your mission. You are then told your score and time taken, and asked if you want to try again. Type "Y" for another game and "N" if you wish to stop — nothing else will be accepted.

If the computer has been expanded and so has extra memory you could use the user-definable graphics capability of the Vic to improve the game. If so, the following routine should be added at the end of the program, and line 3 changed to:

```
3 PRINT CHR$(14): GOSUB 1000
1000 FOR I = 0 TO 1024
1010 POKE 5120 + I, PEEK (32768 + I): NEXT I
1020 FOR I = 0 TO 1024 : READ A
1030 IF A = 1 THEN 1070
1040 POKE 6144 + I, A : NEXT
1050 DATA 56, 124, 230, 3, 3, 230, 124, 56
1060 DATA 24, 60, 102, 231, 166, 24, 36, 68
1070 POKE 36869, 253 : POKE 36866, PEEK
     (36866) OR 128
1080 RETURN
```

If this program is used, the Pokes and Peeks will have to be changed — 60 to 128 and 62 to 129.

The main variables used in the program are:

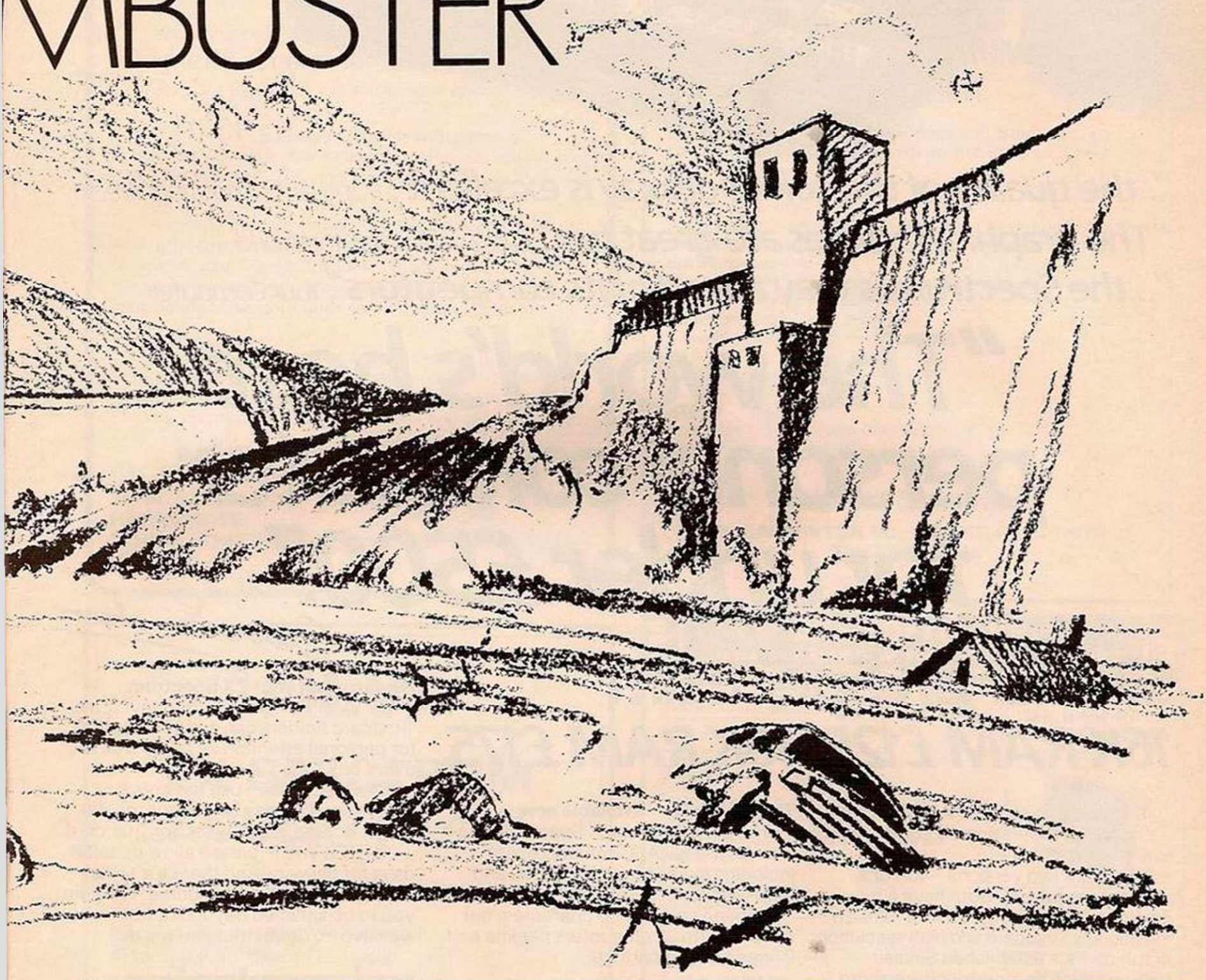
B — position of ship
C — position of bomb
D — position of Nibbler
O — highest score
S — score

The other variables are used mainly in For — Next loops etc. Here is a more detailed explanation of the program:

3:	switches to text mode.
4 to 68:	print the title page. The graphic symbols are obtained by pressing shift and each of the following: N I B B L E R S .
65:	note that there are eight spaces after the cursor symbols.
70 to 150:	print out the instructions for the game.
70:	the graphics symbol is shift and "T".
80:	the graphics symbol is shift and "N".
110:	the graphics symbol is shift and "T".
120:	the graphics symbols are shift, together with "S" and then "N".
130:	the graphics symbol is shift and "I".
140:	the graphics symbol is shift and "H".
159 to 260:	construct the dam and fill the reservoir with water.
261 to 262:	print the score and resets the timer.
270 to 280:	set up the random position of the Nibbler.
290 to 294:	check to see if the Nibbler is hitting the dam or the water.
295:	turns on the sound register and vibrates screen from left to right.
450:	if the space bar is pressed then the bomb drops, else continue moving Nibbler.
491 to 500:	plot the falling bomb.
505 to 512:	checks to see if bomb has hit the Nibbler or the ground. If not, then continue to plot the bomb.
552 to 554:	explosion sound effect.
565 to 568:	vibrate screen up and down.
578:	restores screen to normal position.
590 to 616:	plot the water pouring out of the dam.
620 to 646:	print your score and the time that you lasted for.
620:	graphics symbol is shift and "T".
645:	graphics symbol is shift and "Y".
650:	graphics symbol is shift and "A". Type "Y" for another go, or "N" if you wish to stop.
710:	switches the computer back into graphics mode.

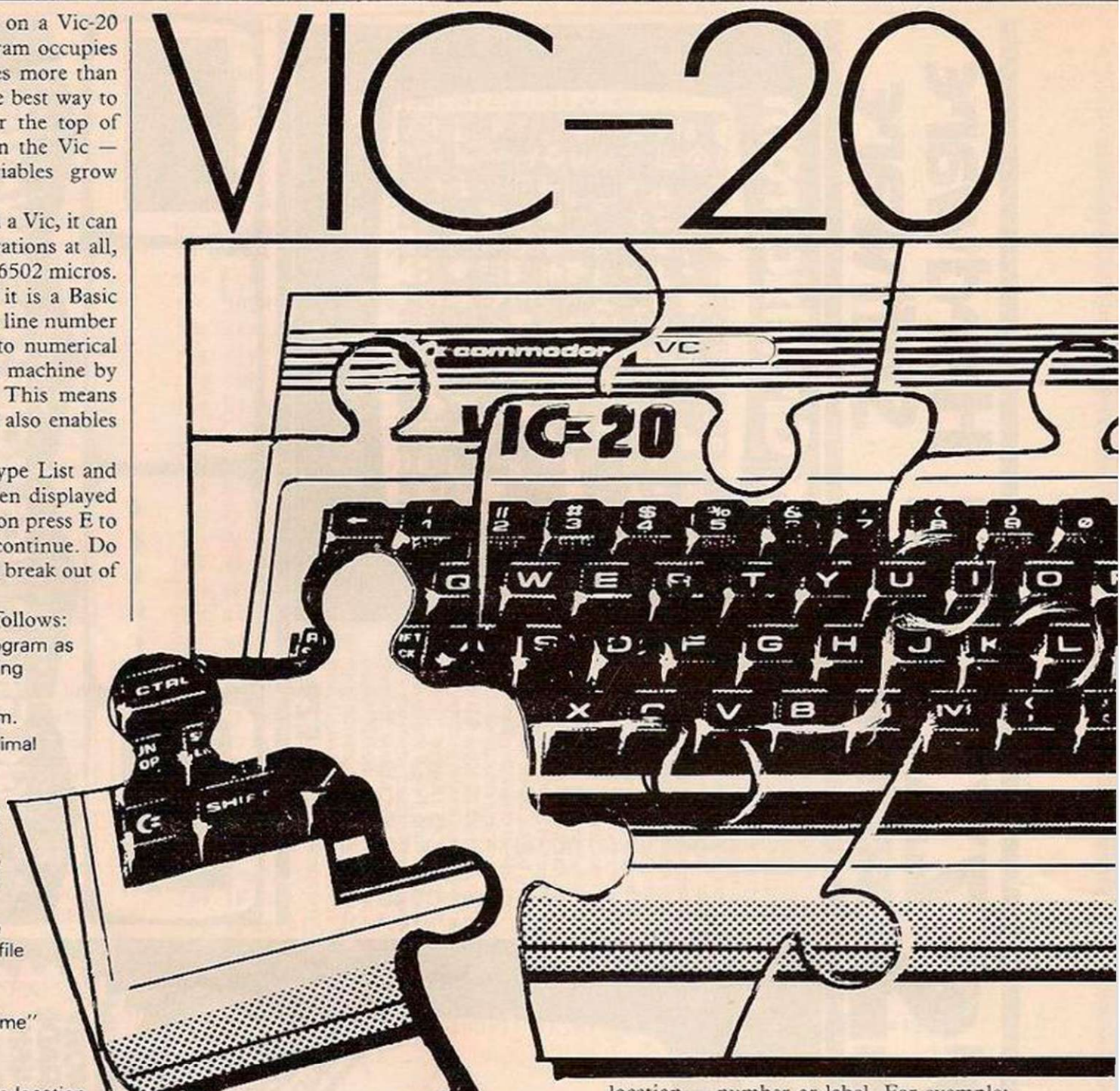
```
0 REM    NIBBLERS              BY D.SHAMBROOK
1 REMCONCAT
2 PRINT"J":O=-32
3 PRINTCHR$(14)
4 POKE36879,237:POKE36878,15
5 FORM=7680TO7701:POKEM,224:NEXT
6 FORM=7702TO8164STEP22:POKEM,224:POKEM+21,224:NEXT
7 FORM=8164TO8185:POKEM,224:NEXT
8 FORB=1TO5
9 POKE36875,200:POKE36876,0
10 FORL=1TO500:NEXT
15 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▶▶▶▶▶▶ ";:FORP=1TO200:NEXT
20 PRINT"\";:FORP=1TO200:NEXT
25 PRINT"|";:FORP=1TO200:NEXT
27 PRINT"|";:FORP=1TO200:NEXT
30 PRINT"L";:POKE36876,200:POKE36875,0:FORP=1TO200:NEXT
35 PRINT"‾";:FORP=1TO200:NEXT
40 PRINT"_";:FORP=1TO200:NEXT
50 PRINT"♥";:FORP=1TO200:NEXT
65 PRINT"▮▮▮▮▮▮▮▮▮        ";:FORP=1TO200:NEXT
68 NEXT
69 POKE36879,25:POKE36876,0
70 PRINT"J HE OBJECT OF THE GAME"
80 PRINT"IS TO HIT THE /IBBLERS"
90 PRINT"BEFORE THEY DESTROY":PRINT
100 PRINT"THE DAM.":PRINT
110 PRINT"   |HE CONTROLS ARE:":PRINT
120 PRINT"♥PACE TO STOP /IBBLER":PRINT:PRINT"AND LAUNCH YOUR SHIP,":PRINT
130 PRINT"AND SPACE AGAIN TO":PRINT:PRINT"DROP YOUR BOMB.\F YOU "
131 PRINT"MISS YOU LOSE A POINT.":PRINT
140 PRINT"|IT A KEY TO CONTINUE"
150 GETA$:IFA$=""THEN150
151 POKE36879,191
155 PRINT"J"
156 FORM=8164TO8185:POKEM,224:POKEM+30720,5:NEXT
159 REM****************         *CONSTRUCT DAM*         ****************
165 V=12:POKE36878,15
170 A=0
180 FORB=7724TO8164STEP22
185 POKE36877,220:FORX=1TO5:NEXT:POKE36877,0
190 POKEA+B,224:POKEA+B+30720,6
200 NEXTB
210 A=A+1:IFA<2THEN180
220 FORA=0TO1
230 FORB=7726TO8167STEP22
235 POKE36877,220:FORX=1TO5:NEXT:POKE36877,0
240 POKEA+B,102:POKEA+B+30720,0
250 NEXTB
260 NEXTA
261 PRINT"&SCORE";S
262 TI$="000000"
263 PRINT"&▶▶▶▶▶▶▶▶▶▶▶▶HIGH"O
264 REM****************         ****NIBBLER****         ****************
270 A=INT(RND(1)*19)+1:D=7744:B$="":F=0
280 FORP=1TOA:D=D+22:NEXT
290 IF PEEK(D)<>102THENR=0:GOTO380
294 IFPEEK(D-1)=224THEN590
295 POKE36878,15:POKE36864,11:POKE36877,250:FORM=1TO40:NEXT:POKE36877,0:POKED,3
2
300 POKE36864,12:GOTO270
380 POKED,60:POKED+30720,0
385 IFB$=" "THEN400
390 GETB$:IFB$=" "THEN430
400 D=D-1
```

MBUSTER



```
405 POKE36878,15:POKE36876,140:FORW=1TO20:NEXT:POKE36876,0
410 FORM=1TO20:NEXT
420 POKED+1,32:GOTO290
424 REM**************        ****SHIP*******        **************
430 FORB=7702TO7723
440 POKEB,62:POKEB+30720,0
445 POKE36878,15:POKE36876,220:FORP=1TO5:NEXTP:POKE36876,0
450 GETC$:IFC$="" "THEN490
470 FORP=1TO20:NEXTP
480 POKEB,32:NEXTB
481 GOTO380
484 REM**************        ****DOMD*******        **************
490 L=200
491 FORC=8+22TO8+484STEP22
495 IFPEEK(C)=70THEN530
496 POKE36878,5:L=L-1:POKE36876,L
500 POKEC,46:POKEC+30720,0
505 IFPEEK(C+22)=60THENS=S+1:GOTO550
510 R=0
512 IFPEEK(C+44)=224THENPOKE36876,0:POKEC,32:R=1:S=S-1:GOTO550
520 FORP=1TO30:NEXT:POKEC,32
530 NEXT
540 POKEB,32:GOTO380
550 POKE36876,0:POKEC+22,32:POKEC,32
551 PRINT"■SCORE"S
552 POKE36877,220
553 FORW=15TO0STEP-1
554 POKE36878,W
565 IFR=1THENPOKE36865,37
566 FORE=1TO30:NEXTE:IFR=1THENPOKE36865,39
567 FORE=1TO30:NEXTE,W
568 POKE36877,0:IFR=1THENPOKE36865,38
569 FORM=8TO7723
570 POKEM,62:POKEM+30720,0:POKE36878,15:POKE36876,220:FORW=1TO5:NEXT:POKE36876,
0
571 FORH=1TO40:NEXT:POKEM,32
575 NEXT
578 IFR=1THENPOKE36865,38:GOTO290
580 GOTO270
590 FORZ=0TO1:POKED+Z,224:POKED+Z+30720,6:NEXT:PRINTCHR$(142):POKED+2,223:POKED
30722,6
594 POKE36878,4:POKE36877,180
595 FORZ=D+24TO8169STEP22:POKEZ,224:POKEZ+30720,6:NEXT
599 FORZ=8185TO8STEP-1
600 POKEZ,224:POKEZ+30720,6
610 NEXT
620 PRINTCHR$(14);"■■■■■■■■■■IE DAM HAS BEEN":PRINT
630 PRINT"DESTROYED AND ITS ALL":PRINT
640 PRINT"■■■■YOUR FAULT!!!!":PRINT
644 U=INT((TI+0.5)/60)
645 PRINT"■■YOU LASTED"U"■■ SECS":PRINT
646 PRINT"■■■■AND SCORED"S:PRINT:PRINT
647 IFS<0THENS=0
650 PRINT"■■■ANOTHER GO(Y/N)?":PRINT"■"
655 POKE36877,0
660 GETD$
670 IFD$="Y"THENPRINT"□":S=0:GOTO70
680 IFD$<>"N"THEN660
700 PRINT"□"
710 PRINTCHR$(142):POKE650,0
720 END
```

VIC-20



THIS ASSEMBLER was developed on a Vic-20 with 16K RAM pack. The program occupies slightly less than 9K but requires more than 13K to run. I have found that the best way to store the assembled code is near the top of memory — locations 53 to 56 on the Vic — Poked low because string variables grow downwards and can interfere.

Although it has been written on a Vic, it can be entered on a Pet with no alterations at all, and with little alteration on most 6502 micros. The source code is entered as if it is a Basic program — that is, each line has a line number and the assembler sorts these into numerical order. The lines are input to the machine by opening a file to the keyboard. This means that there is no "?" prompt, and also enables screen editing.

To list your whole program, type List and press Return. The program is then displayed 15 lines at a time. After each section press E to stop listing, or any other key to continue. Do not press the Stop key as you will break out of the assembler, not the listing.

Other direct commands are as follows:

LISTx	will display the program as above, but beginning at line x.
NEW	clears your program.
ASSEMBLE	displays a hexadecimal assembly of your program.
ASSEMBLEM	as above, but also loads the machine code into memory, as specified within the source code.
SAVE "program name"	outputs the source code to tape as a file named "program name".
LOAD "program name"	loads "program name" from tape.
*SAVE "program name" x-y	saves memory from location x to location y and names it "program name".
*LOAD "program name"	loads "program name" into memory, returning start and end addresses.
DISASSx	disassembles from location x, codes as for the assembler. One screen is displayed at a time. Hit E to end or any other key to continue.
?Hx	returns the hexadecimal value of the denary number x.
?Dx	returns denary value of hexadecimal x.
END	exits the assembler.

Commands are not altogether standard, and spaces are very important — they enable the main assembler routines to split each line into its different sections. Necessary spaces are marked here as [s]. Numbers can be entered in three different forms — as labels, denary or hexadecimal numbers. Labels are preceded by a full stop, and hexadecimal numbers by a $. Labels can be defined as follows:

```
10DL[s]SCREEN[s]4096
```

This defines a label called Screen, and sets it equal to the denary number 4096. Alternatively:

```
20 .LOOP[s]STA[s] .SCREEN
```

On reaching this line, the label Loop is set equal to the location of the command following — the Sta command. Also, the label Screen would on assembly by 4096. be substituted

You are not confined to calculating your own location values. Up to 10 numbers or can follow the commands. For example:

```
20 .LOOP[s]STA[s] .SCREEN+$FF-200
```

This will be calculated by the assembler to give 4151 denary.

To enter a series of letters or graphics, the Byt command is used. This has two alternatives:

```
10 BYT 'THIS IS A TEST
```

The apostrophe before "This" tells the assembler to use ASCII codes.

```
20 BYT PTHIS IS A TEST
```

The P tells the assembler to use CBM screen codes. Branching can be done to either a label or a specified location.

To specify the load location the "*=" command is used. For example,

```
10 *= s 675
```

continues assembly from 675 denary.

```
50 *= s $fff
```

continues assembly from fff hexadecimal. The last line of the source code must be an End command, otherwise the program will loop indefinitely.

For absolute addressing the mnemonic is typed, followed by a space and then the location — number or label. For example:

```
10 LDA[s]SCREEN
```

In immediate addressing, the mnemonic is typed, followed by a #, a space, and then the number or label. For example:

```
10 LDX# [s]200
```

For absolute indexed addressing, the index register to be used is placed immediately after the mnemonic, then a space, then the location. For example:

```
10 LDAX[s]1000
20 INCY[s].LOC
```

With zero-page addressing, a Z follows the mnemonic, before any index register:

```
10 LDAZ[s]100
20 DECZX[s]100
```

In indirect addressing, the index register required is placed in brackets after the mnemonic:

```
10 LDA(Y)[s]100
20 STA(X)[s]150
```

An indirect jump has an I in brackets:

```
10 JMP(I)[s]2000
```

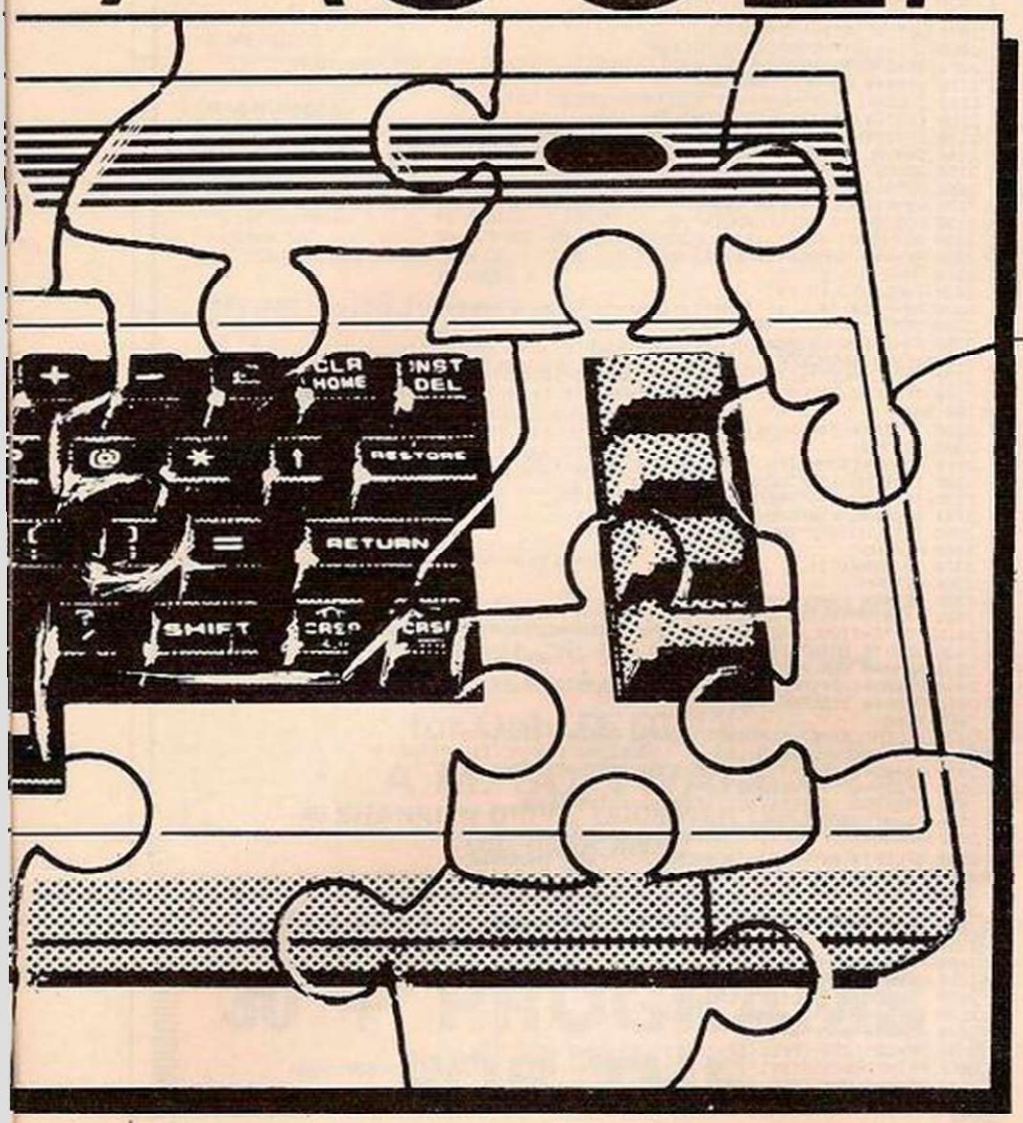
For accumulator addressing, the shift and rotate instructions to be carried out on the accumulator are followed by an A:

```
10 LSRA
20 ROLA
```

Here are the main sections of the program:

10 to 70	read mnemonic data and number of bytes per command.
100 to 440	run through your program and supply each line with its memory

ASSEMBLER



This 6502 assembler, written by Philip Horton, is not solely applicable to Vic machines. Numbers can be entered as denary, hex numbers, or as labels — and you are not confined to calculating your own location values.

```
10 CLR:PRINT"□":DIML$(200),A%(255),LA$(50),P(50),D%(200),C$(200)
20 DIMA$(255):FORJ=0TO254:READA$(J):NEXT
30 FORJ=0TO255:IFA$(J)<>""THENREADA%(J)
40 NEXT
50 PRINT"□■    6502 ASSEMBLER"
60 PRINT"          ─────────"
70 PRINT"(C) P.J. HORTON 1982■"
80 GOTO1410 REM*** INPUT ROUTINE ***
90 J=0
100 REM*** SETUP FOR ASSEMBLY ***
110 J=J+1:C=0:D%(J)=D
120 IFLEFT$(L$(J),1)="."THEN370
130 IFL$(J)="END"THEN450
140 C=C+1:IFC>LEN(L$(J))THEN190
150 IFMID$(L$(J),C,1)<>" "THENC$(J)=C$(J)+MID$(L$(J),C,1):GOTO140
160 IFC$(J)="BYT"THENGOSUB1050:GOTO110
170 IFC$(J)="*="THENGOSUB230:GOTO110
180 IFC$(J)="DL"THENGOSUB290:GOTO110
190 FORF=0TO255
200 IFC$(J)<>A$(F)THENNEXT:PRINT:PRINT"?SYNTAX ERROR IN";:ER=1:GOTO1900
210 D=D+A%(F)
220 GOTO110
230 REM*** NEW MEM LOC ***
240 FORK=1TOLEN(L$(J)):IFMID$(L$(J),K,1)<>" "THENNEXT:PRINT"?SYNTAX ERROR IN";
ER=1:GOTO1900
250 T$=RIGHT$(L$(J),LEN(L$(J))-K)
260 C$(J)="M"
270 GOSUB2200
280 D=VAL(T$):AD=D:RETURN
290 REM*** DEF LABEL ***
300 NL=NL+1:C=3
310 C=C+1:IFMID$(L$(J),C,1)<>" "THENLA$(NL)=LA$(NL)+MID$(L$(J),C,1):GOTO310
320 T$=RIGHT$(L$(J),LEN(L$(J))-C):GOSUB2200:P(NL)=VAL(T$)
330 IFNL=1THEN360
340 FORF=1TONL-1:IFLA$(F)=LA$(NL)THENPRINT"■LABEL REPEATED IN";:ER=1:GOTO1900
350 NEXT
360 RETURN
370 REM*** LABEL ***
380 NL=NL+1:C=1
390 C=C+1:IFMID$(L$(J),C,1)<>" "THENLA$(NL)=LA$(NL)+MID$(L$(J),C,1):GOTO390
400 IFNL=1THEN430
410 FORF=1TONL-1:IFLA$(F)=LA$(NL)THENPRINT"■LABEL REPEATED IN";:ER=1:GOTO1900
420 NEXT
430 P(NL)=D
440 GOTO140
450 REM*** ASSEMBLE ***
460 PRINT"■  LOC        CODE":SC=0
470 N=0
480 PRINT:FORF=1TOJ-1
490 IFC$(F)="M"THENAD=D%(F+1):GOTO700
500 IFC$(F)="DL"THEN700
510 IFC$(F)="B"THENGOSUB1090:GOTO690
520 PRINT:PRINTD%(F),:N=N+1
530 FORK=0TO255:IFA$(K)<>C$(F)THENNEXT
540 DN=K:GOSUB740
550 PRINTTAB(12);H$;
560 IFLA=1THENGOSUB1350
570 IFA%(K)=1THENPRINT:GOTO690
580 IFLEFT$(C$(F),1)="B"ANDC$(F)<>"BIT"THENGOSUB850:GOTO690
590 IFA%(K)=2THENGOSUB940:GOTO690
600 REM*** 2 BYTE JMP ETC. ***
610 FORK=LEN(L$(F))TO1STEP-1:IFMID$(L$(F),K,1)<>" "THENNEXT
620 T$=RIGHT$(L$(F),LEN(L$(F))-K)
630 GOSUB2200
640 T=VAL(T$):D1=INT(T/256):D2=T-D1*256
650 DN=D2:GOSUB740:PRINTTAB(15);H$;
660 IFLA=1THENGOSUB1350
```

(continued on next page)

location, and also set up any labels — either defined or from a full-stop command.

450 to 1400 assemble your program, into memory if required. These lines include:

740 to 840 denary to hex converter for numbers up to 65535 — FFFF.

1010 to 1040 hex to denary converter.

1290 this line converts a letter or graphic into CBM screen Poke codes.

1410 to 2190 contain the input routine which accesses all other parts of the assembler.

2200 to 2380 one subroutine used by all parts of the program to convert the string T$ to a denary number. It is returned as T$.

2450 to 2660 are where the disassembler is accessed by the input routine.

2660 is where the CHR$ commands open inverted commas and then delete them. This prevents, for example, a clear screen command from interfering with any printout.

2670 to 2890 are the subroutines which output memory to tape and vice versa.

2900 to 3150 hold mnemonic data and data to give the number of bytes required by each command.

Now here are the variables:

A$(n) 6502 mnemonics.

A%(n) number of bytes required by each of the commands.

L$(n) each line of your program.

C$(n) the command on each line of your program.

D%(n) the location of each line of your program in memory.

Each of the last three arrays are Dimensioned to 200 elements, but this can be altered as required, by changing line 10.

NL the number of labels so far encountered.

LA$(n) the names of the above labels.

P(n) the location in memory of each of the above labels.

The label arrays are Dimensioned to 50 elements, but can also be altered as required at line 10.

L$ the line that the input routine is currently working on.

N the number of lines in your program.

DN denary number for conversions.

H$ hex number for conversions.

LA flag used to give "Load on assembly".

C counter used to control the position in the line of program that the assembler is dealing with.

ER error report flag, to give a line number after an error in the source code.

SC the number of lines so far printed on the screen for List, Disass, etc.

Finally, here are the error reports.

SYNTAX ERROR	A direct command has not been understood.
SYNTAX ERROR IN x	There is a mistake in line x of the source code.
LABEL REPEATED IN x	The label defined in line x has already been used in the code.
NO SUCH LABEL IN x	The label mentioned in line x has not been defined.
NO CODE ERROR	The user has tried assembling with no code in the memory.

(continued from previous page)

```
670 IN=D1:GOSUB740:PRINTTAB(18);H$
680 IFLA=1THENGOSUB1350
690 SC=SC+1:IFSC=11THENGOSUB720
700 NEXTF:GOTO1380
710 REM*** SCREEN FULL ***
720 GETZ$:IFZ$=""THEN720
730 PRINT"Z":SC=0:RETURN
740 REM*** DEC-HEX ***
750 N1=INT(IN/4096)
760 D2=IN-N1*4096:N2=INT(D2/256)
770 I2=D2-N2*256:N3=INT(D2/16)
780 D2=D2-N3*16:N4=D2
790 H$=CHR$(N1+48-(N1>9)*7)
800 H$=H$+CHR$(N2+48-(N2>9)*7)
810 H$=H$+CHR$(N3+48-(N3>9)*7)
820 H$=H$+CHR$(N4+48-(N4>9)*7)
830 IFLEFT$(H$,1)="0"ANDLEN(H$)>2THENH$=RIGHT$(H$,LEN(H$)-1):GOTO830
840 RETURN
850 REM*** BRANCH ***
860 FORK=LEN(L$(F))TO1STEP-1:IFMID$(L$(F),K,1)<>" "THENNEXT
870 T$=RIGHT$(L$(F),LEN(L$(F))-K)
880 GOSUB2200
890 IN=VAL(T$):T=IX(F):IFIN<TTHENIN=IN-T-2:GOTO910
900 IN=255-(T-IN+1)
910 GOSUB740:PRINTTAB(15);H$
920 IFLA=1THENGOSUB1350
930 RETURN
940 REM*** 1 BYTE JMP ETC. ***
950 FORK=LEN(L$(F))TO1STEP-1:IFMID$(L$(F),K,1)<>" "THENNEXT
960 T$=RIGHT$(L$(F),LEN(L$(F))-K)
970 GOSUB2200
980 IN=VAL(T$):GOSUB740:PRINTTAB(15);H$
990 IFLA=1THENGOSUB1350
1000 RETURN
1010 REM*** HEX-DEC ***
1020 IN=0:FORH=1TOLEN(H$)
1030 IN=IN+(ASC(MID$(H$,H,1))-48+(ASC(MID$(H$,H,1))>57)*7)*(16^(LEN(H$)-H)):NEX
TH
1040 RETURN
1050 REM*** BYT ***
1060 FORK=LEN(L$(J))TO1STEP-1:IFMID$(L$(J),K,1)<>","ANDMID$(L$(J),K,1)<>"B"THEN
NEXT:GOTO1080
1070 D=D+LEN(L$(J))-K:D$(J)="B":RETURN
1080 PRINT:PRINT"?SYNTAX ERROR IN";ER=1:GOTO1900
1090 REM*** ASSEMBLE BYT ***
1100 FORJ=LEN(L$(F))TO1STEP-1:IFMID$(L$(F),J,1)="/"THEN1130
1110 IFMID$(L$(F),J,1)="B"THEN1250
1120 NEXT:GOTO1200
1130 B$=RIGHT$(L$(F),LEN(L$(F))-J)
1140 FORJ=1TOLEN(B$):PRINT
1150 PRINTD%(F)+J-1;:IN=ASC(MID$(B$,J,1)):GOSUB740:PRINTTAB(12);H$
1160 IFTP=1THENPRINT#1,H$
1170 IFLA=1THENGOSUB1350
1180 SC=SC+1:IFSC=11THENGOSUB720
1190 NEXT:RETURN
1200 REM*** NUMBER ***
1210 T$=RIGHT$(L$(F),LEN(L$(F))-J)
1220 GOSUB2200
1230 IN=VAL(T$):GOSUB740:PRINTTAB(12);H$
1240 IFLA=1THENGOSUB1350
1250 REM*** POKE ALPHA ***
1260 B$=RIGHT$(L$(F),LEN(L$(F))-J)
1270 FORJ=1TOLEN(B$):PRINT
1280 PRINTD%(F)+J-1;TAB(12);
1290 IN=(ASC(MID$(B$,J,1))AND128)/20R(ASC(MID$(B$,J,1))AND63)
1300 GOSUB740:PRINTTAB(12);H$
1310 IFTP=1THENPRINT#1,H$
1320 IFLA=1THENGOSUB1350
1330 SC=SC+1:IFSC=11THENGOSUB720
1340 NEXT:RETURN
1350 REM*** ASSEMBLER LOADER ***
1360 IFIN>255THENPRINT:PRINT"NUMBER TOO LARGE IN";:ER=1:GOTO1900
1370 POKEAD,IN:AD=AD+1:RETURN
1380 REM*** END ***
1390 IFTP=1THENPRINT#1,"END":CLOSE1
1400 GOSUB720:GOTO1900
1410 REM*** INPUT ***
1420 REM
1430 OPEN2,0
1440 PRINT"READY."
1450 INPUT#2,L$:PRINT
1460 IFL$="NEW"THENFORJ=1TON:L$(J)="":NEXT:N=0:GOTO1440
1470 IFLEFT$(L$,4)="LIST"THEN1980
1480 IFL$="END"THENPRINT"Z":END
1490 IFLEFT$(L$,6)="DISASS"THEN2450
1500 IFLEFT$(L$,1)="?"THEN2100
1510 IFLEFT$(L$,8)="ASSEMBLE"THEN1800
1520 IFLEFT$(L$,4)="LOAD"THEN1700
1530 IFLEFT$(L$,5)="*LOAD"THEN2790
1540 IFLEFT$(L$,4)="SAVE"THEN1750
1550 IFLEFT$(L$,5)="*SAVE"THEN2670
1560 N1=VAL(L$):IFN1=0THENPRINT:PRINT"?SYNTAX ERROR":GOTO1450
1570 IFLEN(STR$(VAL(L$)))-1=LEN(L$)THEN1650
1580 N=N+1:IFN=1THENL$(1)=L$:GOTO1450
1590 FORJ=1TON-1:IFVAL(L$(J))<>N1THENNEXT
1600 IFVAL(L$(J))=N1THENL$(J)=L$:N=N-1:GOTO1450
1610 FORK=NTOJSTEP-1
1620 L$(K)=L$(K-1):NEXT
1630 L$(J)=L$
1640 GOTO1450
1650 REM*** SPLAT LINE ***
1660 FORK=1TON:IFVAL(L$(K))<>VAL(L$)THENNEXT:GOTO1450
1670 FORL=KTON:L$(L)=L$(L+1):NEXT:N=N-1
1680 IFN<0THENN=0
1690 GOTO1450
1700 REM*** LOAD ***
1710 OPEN1,1,0,RIGHT$(L$,LEN(L$)-4)
1720 INPUT#1,N:FORJ=1TON
1730 INPUT#1,L$(J):NEXT
1740 CLOSE1:PRINT:PRINT"READY.":GOTO1450
1750 REM*** SAVE ***
1760 OPEN1,1,1,RIGHT$(L$,LEN(L$)-4)
1770 PRINT#1,N:FORJ=1TON
1780 PRINT#1,L$(J):NEXT
1790 CLOSE1:PRINT:PRINT"READY.":GOTO1450
1800 REM*** ENTERED ***
1810 LA=0
1820 IFRIGHT$(L$,1)="M"THENLA=1
1830 IFN=0THENPRINT"NO CODE ERROR":GOTO1440
1840 PRINT:PRINT"WORKING..."
1850 FORJ=1TON
1860 NC=VAL(L$(J)):NC=LEN(STR$(NC))+1
1870 L$(J)=RIGHT$(L$(J),LEN(L$(J))-NC)
1880 IFLEFT$(L$(J),1)=" "THENL$(J)=RIGHT$(L$(J),LEN(L$(J))-1):GOTO1880
1890 NEXTJ:SN=N:GOTO90
1900 REM*** RENUM&CONT ***
1910 IFER=1THENER=0:PRINT10*J
1920 IFE2=1THENE2=0:PRINT10*F
1930 N=SN:FORJ=1TON:L$(J)=STR$(10*J)+" "+L$(J):D$(J)="
1940 L$(J)=RIGHT$(L$(J),LEN(L$(J))-1):NEXT
1950 FORJ=1TOPL:LA$(J)="":P(J)=0:NEXT:NL=0
1960 PRINT"K"
1970 GOTO1440
1980 REM*** LIST ***
1990 SC=0
2000 IFL$="LIST"THENSL=1:GOTO2040
2010 SL=VAL(RIGHT$(L$,LEN(L$)-4))
2020 FORJ=1TON:IFSL>VAL(L$(J))THENNEXT:GOTO1440
2030 SL=J
2040 FORJ=SLTON:PRINTL$(J)
2050 SC=SC+1:IFSC=15THEN2070
2060 NEXT:GOTO1440
2070 GETZ$:IFZ$=""THEN2070
2080 IFZ$="E"THENPRINT:GOTO1440
2090 SC=0:GOTO2060
2100 REM*** CONVERSIONS ***
2110 IFLEN(L$)<3THENPRINT"K?SYNTAX ERROR":GOTO1440
2120 IFMID$(L$,2,1)="D"THEN2170
2130 IFMID$(L$,2,1)<>"H"THENPRINT"K?SYNTAX ERROR":GOTO1440
2140 IN=VAL(RIGHT$(L$,LEN(L$)-2))
2150 GOSUB740:PRINT"KHEX=";H$
2160 GOTO1440
2170 H$=RIGHT$(L$,LEN(L$)-2)
2180 GOSUB1010:PRINT"KDEC=";IN
2190 GOTO1440
2200 REM*** NUMBER T$ ***
2210 NN=0
2220 FORL=2TOLEN(T$)
2230 A$=MID$(T$,L,1):IFA$="+"ORA$="-"ORA$="*"ORA$="/"THEN2440
2240 NEXTL:NN=NN+1:T$(NN)=T$
2250 FI=0:FORNU=1TONN
2260 IFNU=1THEN2280
2270 S$(NU)=LEFT$(T$(NU),1):T$(NU)=RIGHT$(T$(NU),LEN(T$(NU))-1)
2280 IFLEFT$(T$(NU),1)="$"THENH$=RIGHT$(T$(NU),LEN(T$(NU))-1):GOSUB1010:T$(NU)=
STR$(IN)
2290 IFLEFT$(T$(NU),1)=","THEN2400
2300 NEXTNU
2310 FORNU=1TONN:IFNU=1THENFI=VAL(T$(1)):GOTO2360
2320 IFS$(NU)="+"THENFI=FI+VAL(T$(NU))
2330 IFS$(NU)="-"THENFI=FI-VAL(T$(NU))
2340 IFS$(NU)="*"THENFI=FI*VAL(T$(NU))
2350 IFS$(NU)="/"THENFI=FI/VAL(T$(NU))
2360 NEXTNU
2370 T$=STR$(FI)
2380 RETURN
2390 REM*** LABEL ***
2400 A$=RIGHT$(T$(NU),LEN(T$(NU))-1)
2410 FORTE=1TONL:IFLA$(TE)=A$THENT$(NU)=STR$(P(TE)):GOTO2300
2420 NEXTTE:PRINT:PRINT"NO SUCH LABEL IN";:E2=1:GOTO1900
2430 REM*** NEW NO ***
2440 NN=NN+1:T$(NN)=LEFT$(T$,L-1):T$=RIGHT$(T$,LEN(T$)-L+1):GOTO2220
2450 REM*** DISSASS ***
2460 SC=0
2470 IFLEN(L$)=6THENPRINT"?SYNTAX ERROR":GOTO1440
2480 T$=RIGHT$(L$,LEN(L$)-6):GOSUB2200:D2=VAL(T$):PRINT"Z"
2490 SC=SC+1:IFSC=11THENGOSUB720:IFZ$="E"THEN1440
2500 IFD2>65535THEN63999
2510 PRINTD2;:A=PEEK(D2):IFA$(A)=""THEN2660
2520 PRINTTAB(7)A$(A);
2530 IFA%(A)=1THENPRINT:GOTO2590
2540 IFLEFT$(A$(A),1)="B"ANDA$(A)<>"BRK"THEN2610
2550 IFA%(A)=2THENDN=PEEK(D2+1):D2=D2+2
2560 IFA%(A)=3THENDN=PEEK(D2+1)+(PEEK(D2+2)*256):D2=D2+3
2570 PRINTTAB(13);DN
2580 PRINT:GOTO2490
2590 D2=D2+1
2600 PRINT:GOTO2490
2610 REM*** BRANCH ***
2620 IFPEEK(D2+1)>127THEN2640
2630 DN=D2+PEEK(D2+1)+2:D2=D2+2:GOTO2570
2640 REM*** BACK ***
2650 DN=D2-(255-(PEEK(D2+1)+1)):D2=D2+2:GOTO2570
2660 PRINTTAB(7);"BYT";TAB(14);CHR$(34);CHR$(20);CHR$(A):GOTO2590
2670 REM*** MEM-TAPE ***
2680 FORJ=6TOLEN(L$):IFMID$(L$,J,1)<>CHR$(34)THENNEXT:PRINT"?SYNTAX ERROR":GOTO
1440
2690 NS=J+1
2700 FORJ=NS+1TOLEN(L$):IFMID$(L$,J,1)<>CHR$(34)THENNEXT:PRINT"?SYNTAX ERROR":G
OTO1440
2710 NS=J-1
2720 F$=MID$(L$,NS,NE-NS+1)
2730 FORJ=NE+2TOLEN(L$):IFMID$(L$,J,1)<>"-"THENNEXT:PRINT"?SYNTAX ERROR":GOTO14
40
2740 T$=MID$(L$,NE+2,J-NE-2):GOSUB2200:D2=VAL(T$)
2750 T$=RIGHT$(L$,LEN(L$)-J):GOSUB2200:DE=VAL(T$)
2760 OPEN1,1,1,F$
2770 PRINT#1,D2:PRINT#1,DE:FORJ=D2TODE
2780 N1%=PEEK(J):PRINT#1,N1%:NEXT:CLOSE1:GOTO1440
2790 REM*** TAPE-MEM ***
2800 FORJ=6TOLEN(L$):IFMID$(L$,J,1)<>CHR$(34)THENNEXT:PRINT"?SYNTAX ERROR":GOTO
1440
2810 NS=J+1
2820 FORJ=NS+1TOLEN(L$):IFMID$(L$,J,1)<>CHR$(34)THENNEXT:PRINT"?SYNTAX ERROR":G
OTO1440
2830 NE=J-1
2840 F$=MID$(L$,NS,NE-NS+1)
2850 OPEN1,1,0,F$
2860 INPUT#1,D2:INPUT#1,DE
2870 FORJ=D2TODE:INPUT#1,N1%:POKEJ,N1%:NEXT:CLOSE1
2880 PRINT"KSTART ADDRESS";D2
2890 PRINT"KEND ADDRESS";DE:GOTO1440
2900 REM*** MNEMONIC DATA ***
2910 DATABRK,ORA(X),,,,ORAZ,ASLZ,,PHP,ORA#,ASLA,,,ORA,ASL,,BPL
2920 DATAORA(Y),,,,ORAZX,ASLZX,,CLC,ORAY,,,,ORAX
2930 DATA ASLX,,JSR,AND(X),,,BITZ,ANDZ,ROLZ,,PLP,AND#,ROLA,,BIT,AND,ROL,,BMI
2940 DATAAND(Y),,,,ANDZX,ROLZX,,SEC,ANDY,,,,ANDX
2950 DATAROLX,,RTI,EOR(X),,,,EORZ,LSRZ,,PHA,EOR#,LSRA,,JMP
2960 DATAEOR,LSR,,BVC,EOR(Y),,,,EORZX,LSRZX,,CLI,EORY,,,,EORX
2970 DATALSRX,,RTS,ADC(X),,,,ADCZ,RORZ,,PLA,ADC#,RORA,,JMP(I)
2980 DATAADC,ROR,,BVS,ADC(Y),,,,ADCZX,RORZX,,SEI,ADCY,,,,ADCX
2990 DATAROPX,,,STA(X),,,STYZ,STAZ,STXZ,,DEY,,TYA,,STY,STA
3000 DATASTX,,BCC,STA(Y),,,STYZX,STAZX,STXZY,,TYA,STAY,TXS,,,STAX,,,LDY#
3010 DATALDA(X),LDX#,,LDYZ,LDAZ,LDXZ,,TAY
3020 DATALDA#,TAX,,LDY,LDA,LDX,,BCS
3030 DATALDA(Y),,,LDYZX,LDAZX,LDXZY,,CLV,LDAY,TSX,,LDYX,LDAY,LDXY,,CPY#
3040 DATACMP(X),,,CPYZ,CMPZ,DECZ,,INY,CMP#,DEX,,CPY,CMP,DEC,,BNE
3050 DATACMP(Y),,,,CMPZY,DECZX,,CLD,CMPY,,,,CMPX,DECX,,CPX#
3060 DATASBC(X),,,CPXZ,SBCZ,INCZ,,INX,SBC#,NOP,,CPX,SBC,INC,,BEQ
3070 DATASBC(Y),,,,SBCZX,INCZX,,SED,SBCY,,,,SBCX,INCX
3080 REM*** BYTES DATA ***
3090 DATA1,2,2,2,1,2,1,3,3,2,2,2,2,1,3,3,3,3,2,2,2,2
3100 DATA1,2,1,3,3,3,2,2,2,2,1,3,1,3,1,2,2,2,1,2,1,3,3,3
3110 DATA2,2,2,2,1,3,3,3,1,2,2,2,1,2,1,2,3,3,2,2,2,2,1,3
3120 DATA3,3,2,2,2,2,2,1,1,3,3,3,2,2,2,2,2,1,3,1,2,2,2
3130 DATA2,2,2,1,2,1,3,3,3,2,2,2,2,2,1,3,1,3,3,3,2,2,2,2,2
3140 DATA1,2,1,3,3,3,2,2,2,2,1,3,3,3,2,2,2,2,2,1,2,1,3,3,3,2,2,2,2
3150 DATA1,3,3,3
READY.
```