Martin Görlitz

Disassembler für 8048/8748-Programme

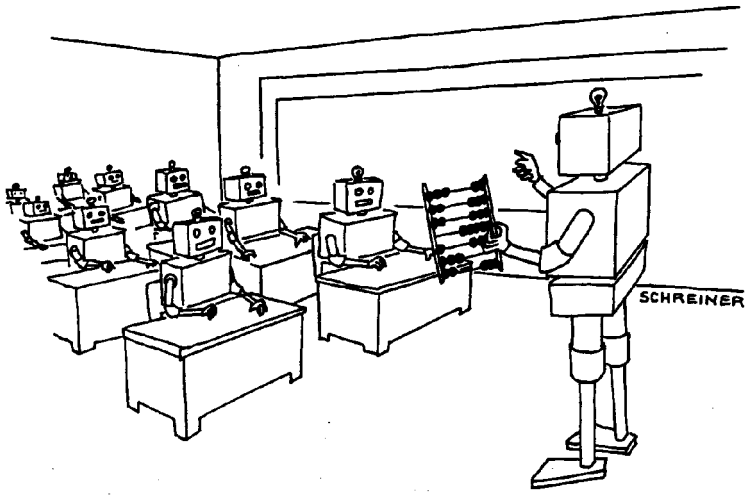
Der folgende Disassembler ist in Basic geschrieben und läuft auf den PET-/CBM-Rechnern, bei Anpassung der Druckerausgabe auch auf allen anderen Basic-Rechnern. Mit ihm kann man Programme der genannten Einchip-Mikroprozessoren rückübersetzen. Diese Prozessoren finden sich in vielen Peripheriegeräten wie z. B. intelligenten Druckern.

Der Befehlssatz der 8048-Familie kennt nur Ein- und Zweibyte-Operationen. Bei vielen Operationen, welche z. B. ein Register ansprechen, ist innerhalb des 1-Byte-Befehls noch das spezielle Register adressiert. Über 70 % der 96 Befehle sind daher nur ein Byte lang. Dies macht den Befehlssatz sehr effizient.
Der Disassembler (*Bild 1*) prüft mit seinen DATA-Anweisungen die Befehle ab, wobei er darauf verzichtet, über Befehlsmasken gleiche Befehle mit integriertem Displacement (angesprochenem Register oder I/O-Port) zusammenzufassen. Dies drückt sich in einer recht einfachen und übersichtlichen Programmstruktur aus, man nimmt jedoch eine etwas geringere Übersetzungsgeschwindigkeit in Kauf.
Eine Reihe von Instruktionen bezieht sich auf die augenblickliche Stelle im Programm. So werden Sprünge immer absolut angegeben, gelten aber meist nur für eine Speicherseite. Ein Beispiel:
217 36 0F JT0 $20F bedeutet: „Jump on Test 0" nach 0F in derselben Speicherseite, hier $200. Hier setzt der Disassembler zum Displacement automatisch die Seite hinzu. Dies geschieht auch richtig in einem Grenzfall: Wenn ein solcher Sprungbefehl am Ende einer Speicherseite (bei xFF) auftritt, so wird, da der Programmzähler nach der Ausführung schon weitergezählt ist, nach einem Speicherplatz in der nächsten Seite gesprungen. Da der Disassembler auch erst seinen eigenen Programmzähler weiterstellt und dann das Displacement prüft, bleibt alles in Ordnung.

Die direkt geprüften Befehle stehen innerhalb der DATA-Statements von Zeile 150 bis 520. Hierauf folgen die 2-Byte-Befehle, wobei der Disassembler jeweils in Zeile 810 prüft, ob das Displacement eine Adresse ($) oder ein Operand (#) ist und gegebenenfalls die tatsächliche Adresse dahinterschreibt. Eine besondere Behandlung verlangen drei Befehlsgruppen: Der unbedingte Sprung JUMP, der Subroutine-Aufruf CALL und der Sprung nach Prüfung eines Akkumulatorbits JUMP ON BIT N. Diese finden sich in den Zeilen 880, 910 und 940. Sie werden tatsächlich über Masken in den Zeilen 770 bis 790 erkannt und verarbeitet.
In *Bild 2* ist ein Musterausdruck des Disassemblers zu sehen, welcher den Beginn des Cocos-Mini-Monitors zeigt (vgl. [2]). Wenn ein Ausdruck gewünscht wird, so muß in Zeile 640 die Adresse des Geräts und in der folgenden Zeile CMD angegeben werden.

Literatur

[1] MCS 48 User Manual, Intel.
[2] Görlitz, M.: Programmentwicklung für Einchip-Mikrocomputer, ELEKTRONIK 1980, Heft 21, S. 71...77.



....heute lernen wir das duale Zahlensystem....

```
100 REM 8748-DISASSEMBLER
110 A$="0123456789ABCDEF"
120 DIMH$(15):FORI=1TO16:H$(I-1)=MID$(A$,I,1):NEXT:GOTO630
130 ØP$=H$(INT(OP/16))+H$(OP-16*INT(OP/16))
140 RETURN
150 DATA00,NOP,02,"OUTL BUS,A",05,EN   I,07,DEC  A
160 DATA08,"INS  A,BUS",09,"IN   A,P1",0A,"IN   A,P2",0C
170 DATA"MOVD A,P4",0D,"MOVD A,P5",0E,"MOVD A,P6",0F,"MOVD A,P7"
180 DATA10,INC  @R0,11,INC  @R1,15,DIS  I,17,INC  A,18,INC  R0,19,INC  R1,1A
190 DATAINC  R2,1B,INC  R3,1C,INC  R4,1D,INC  R5,1E,INC  R6,1F,INC  R7
200 REM***CODES 20...2F***
210 DATA20,"XCH  A,@R0",21,"XCH  A,@R1",25,EN   TCNT I,27,CLR  A,28
220 DATA"XCH  A,R0",29,"XCH  A,R1",2A,"XCH  A,R2",2B,"XCH  A,R3",2C,"XCH  A,R4"

230 DATA2D,"XCH  A,R5",2E,"XCH  A,R6",2F,"XCH  A,R7",30,"XCHD A,@R0",31
240 DATA"XCHD A,@R1",35,DIS  TCNT I,37,CPL  A,39,"OUTL P1,A",3A,"OUTL P2,A"
250 DATA3C,"MOV  P4,A",3D,"MOV  P5,A",3E,"MOV  P6,A",3F,"MOV  P7,A",40
260 DATA"ORL  A,@R0",41,"ORL  A,@R1",42,"MOV  A,T",45,STRT CNT,47,SWAP A,48
270 DATA"ORL  A,R0",49,"ORL  A,R1",4A,"ORL  A,R2",4B,"ORL  A,R3",4C
280 DATA"ORL  A,R4",4D,"ORL  A,R5",4E,"ORL  A,R6",4F,"ORL  A,R7",50
290 DATA"ANL  A,@R0",51,"ANL  A,@R1",55,STRT T,57,DA    A,58,"ANL  A,R0",59
300 DATA"ANL  A,R1",5A,"ANL  A,R2",5B,"ANL  A,R3",5C,"ANL  A,R4",5D
310 DATA"ANL  A,R5",5E,"ANL  A,R6",5F,"ANL  A,R7",60,"ADD  A,@R0",61
320 DATA"ADD  A,@R1",62,"MOV  T,A",65,STOP TCNT,67,RRC  A,68,"ADD  A,R0",69
330 DATA"ADD  A,R1",6A,"ADD  A,R2",6B,"ADD  A,R3",6C,"ADD  A,R4",6D
340 DATA"ADD  A,R5",6E,"ADD  A,R6",6F,"ADD  A,R7",70,"ADDC A,@R0",71
350 DATA"ADDC A,@R1",E3,"MOVP3 A,@A",75,ENTO CLK,77,RR   A,78,"ADDC A,R0",79
360 DATA"ADDC A,R1",7A,"ADDC A,R2",7B,"ADDC A,R3",7C,"ADDC A,R4",7D
370 DATA"ADDC A,R5",7E,"ADDC A,R6",7F,"ADDC A,R7",80,"MOVX A,@R0",81
380 DATA"MOVX A,@R1",83,RET,85,CLR F0,8C,"ORLD P4,A",8D,"ORLD P5,A",8E
390 DATA"ORLD P6,A",8F,"ORLD P7,A",90,"MOV  @R0,A",91,"MOV  @R1,A",93,RETR
400 DATA95,CPL  F0,97,CLR  C,9C,"ANLD P4,A",9D,"ANLD P5,A",9E,"ANLD P6,A",9F
410 DATA"ANLD P7,A",A0,"MOV  @R0,A",A1,"MOV @R1,A",A3,"MOVP A,@A",A5,CLR  F1
420 DATAA7,CPL  C,A8,"MOV  R0,A",A9,"MOV  R1,A",AA,"MOV  R2,A",AB,"MOV  R3,A"
430 DATAAC,"MOV  R4,A"
440 DATAAD,"MOV  R5,A",AE,"MOV  R6,A",AF,"MOV  R7,A",B5,CPL  F1,C5,SEL  RB0
450 DATAC7,"MOV  A,PSW",C8,DEC  R0,C9,DEC  R1,CA,DEC  R2,CB,DEC  R3,CC
460 DATADEC  R4,CD,DEC  R5,CE,DEC  R6,CF,DEC  R7,D0,"XRL  A,@R0",D1
470 DATA"XRL  A,@R1",D5,SEL  RB1,D7,"MOV  PSW,A",D8,"XRL  A,R0",D9
480 DATA"XRL  A,R1",DA,"XRL  A,R2",DB,"XRL  A,R3",DC,"XRL  A,R4",DD
490 DATA"XRL  A,R5",DE,"XRL  A,R6",DF,"XRL  A,R7",E5,SEL  MB0,E7,RL   A,F0
500 DATA"MOV  A,@R0",F1,"MOV  A,@R1",F5,SEL  MB1,F7,RLC  A,F8,"MOV  A,R0"
510 DATAF9,"MOV  A,R1",FA,"MOV  A,R2",FB,"MOV  A,R3",FC,"MOV  A,R4",FD
520 DATA"MOV  A,R5",FE,"MOV  A,R6",FF,"MOV  A,R7"
530 DATA03,"ADD  A,#",13,"ADDC A,#"
540 DATA16,JTF  $,23,"MOV  A,#",26,JNT0 $,36,JT0  $,43,"ORL  A,#",46,JNT1 $
550 DATA53,"ANL  A,#",56,JT1  $,76,JF1  $,86,JNI  $,88,"ORL  BUS,#",89
560 DATA"ORL  P1,#",8A,"ORL  P2,#",96,JNZ  $,98,"ANL  BUS,#",99,"ANL  P1,#"
570 DATA9A,"ANL  P2,#",B0,"MOV  @R0,#",B1,"MOV  @R1,#",B6,JF0  $,B8,"MOV  R0,#"

580 DATABB,"MOV  R1,#",BC,"MOV  R2,#",BD,"MOV  R3,#",BC,"MOV  R4,#",BD
```

Bild 1. Der 8048-Disassembler ist in Basic für Computer der PET- und CBM-Serien geschrieben, läßt sich aber leicht auf andere Basic-Computer übertragen

```
460 DATADEC  R4,CD,DEC  R5,CE,DEC  R6,CF,DEC  R7,D0,"XRL  A,@R0",D1
470 DATA"XRL  A,@R1",D5,SEL  RB1,D7,"MOV  PSW,A",D8,"XRL  A,R0",D9
480 DATA"XRL  A,R1",DA,"XRL  A,R2",DB,"XRL  A,R3",DC,"XRL  A,R4",DD
490 DATA"XRL  A,R5",DE,"XRL  A,R6",DF,"XRL  A,R7",E5,SEL  MB0,E7,RL   A,F0
500 DATA"MOV  A,@R0",F1,"MOV  A,@R1",F5,SEL  MB1,F7,RLC  A,F8,"MOV  A,R0"
510 DATAF9,"MOV  A,R1",FA,"MOV  A,R2",FB,"MOV  A,R3",FC,"MOV  A,R4",FD
520 DATA"MOV  A,R5",FE,"MOV  A,R6",FF,"MOV  A,R7"
530 DATA03,"ADD  A,#",13,"ADDC A,#"
540 DATA16,JTF  $,23,"MOV  A,#",26,JNT0 $,36,JT0  $,43,"ORL  A,#",46,JNT1 $
550 DATA53,"ANL  A,#",56,JT1  $,76,JF1  $,86,JNI  $,88,"ORL  BUS,#",89
560 DATA"ORL  P1,#",8A,"ORL  P2,#",96,JNZ  $,98,"ANL  BUS,#",99,"ANL  P1,#"
570 DATA9A,"ANL  P2,#",B0,"MOV  @R0,#",B1,"MOV  @R1,#",B6,JF0  $,B8,"MOV  R0,#"

580 DATAB9,"MOV  R1,#",BA,"MOV  R2,#",BB,"MOV  R3,#",BC,"MOV  R4,#",BD
590 DATA"MOV  R5,#",BE,"MOV  R6,#",BF,"MOV  R7,#",C6,JZ   $,D3,"XRL  A,#"
600 DATAE6,JNC  $,E8,"DJNZ R0,$",E9,"DJNZ R1,$",EA,"DJNZ R2,$",EB,"DJNZ R3,$"
610 DATAEC,"DJNZ R4,$",ED,"DJNZ R5,$",EE,"DJNZ R6,$",EF,"DJNZ R7,$",F6,JC   $
620 REM
630 INPUT"DISASSEMBLIEREN AB DEZ.";A
640 OPEN1,17
650 REM HIER 'CMD', WENN DRUCKEN!
660 REM
670 FORPA=0TO24
680 OP=INT(A/256)AND15:PRINTH$(OP);:OP=A-INT(A/256)*256:GOSUB130:PRINTOP$;
690 PRINT"   ";:OP=PEEK(A):GOSUB130:PRINTOP$;" ";:I=0:RESTORE
700 I=I+1:READA$:IFA$=OP$THENPRINT"       ";:READA$:PRINTA$:GOTO730
710 IFI>324THEN770
720 GOTO700
730 I=0:A=A+1:RESTORE:NEXTPA
740 GETA$:IFA$=""THEN740
750 IFA$=" "THEN650
760 END
770 IF(OP AND31)=4THEN880
780 IF(OP AND31)=20THEN910
790 IF(OP AND31)=18THEN940
800 I=I+1:READA$:IFA$<>OP$THEN840
810 GOSUB980:READA$:PRINTA$;:IFRIGHT$(A$,1)<>"$"THEN830
820 PRINTH$(C);
830 PRINTOP$:A=A+1:GOTO730
840 IFI>408THEN870
850 GOTO800
860 END
870 PRINT"==UNBEKANNT==":GOTO730
880 REM *** JUMP
890 B=(OP AND224)/32
900 GOSUB980:PRINT"JMP  $";H$(B);OP$:A=A+1:GOTO730
910 REM *** CALL
920 B=(OP AND224)/32
930 GOSUB980:PRINT"CALL $";H$(B);OP$:A=A+1:GOTO730
940 REM***JUMP ON BIT N***
950 B=(OP AND224)/32
960 GOSUB980:PRINT"JB";H$(B);"  $";H$(C);OP$:A=A+1:GOTO730
970 END
980 C=((A+1)/256)AND15
990 OP=PEEK(A+1):GOSUB130:PRINTOP$;"     ";:RETURN
READY.
```

```
000   D5        SEL  RB1
001   04 0A     JMP  $00A
003   FF        MOV  A,R7
004   FF        MOV  A,R7
005   FF        MOV  A,R7
006   FF        MOV  A,R7
007   FF        MOV  A,R7
008   FF        MOV  A,R7
009   FF        MOV  A,R7
00A   F4 A1     CALL $7A1
00C   B8 20     MOV  R0,#20
00E   B0 00     MOV  @R0,#00
010   18        INC  R0
011   B0 00     MOV  @R0,#00
013   F4 B6     CALL $7B6
015   F4 74     CALL $774
017   AF        MOV  R7,A
018   C6 13     JZ   $013
01A   03 E0     ADD  A,#E0
01C   96 24     JNZ  $024
01E   F4 23     CALL $723
020   F4 2E     CALL $72E
022   04 0C     JMP  $00C
024   FF        MOV  A,R7
025   03 F3     ADD  A,#F3
```

Bild 2. Musterausdruck eines 8048-Programms in disassemblierter Form

```
100 REM 8048-DISASSEMBLER
110 A$="0123456789ABCDEF"
120 DIM H$(15):FOR I=1 TO 16:H$(I-1)=MID$(A$,I,1):NEXT:GOTO 630
130 OP$=H$(INT(OP/16))+H$(OP-16*INT(OP/16))
140 RETURN
150 DATA 00,NOP,02,"OUTL BUS/A",05/EN  I,07,DEC A
160 DATA 08,"INS    A,BUS"09,"IN      A,P1",OA,"IN      A,P2" OC
170 DATA "MOVD A,P4fl,OD,flMOVD A/P5"/OE/"MOVD A,P6",OF/'MOVD A,P7"
180 DATA 10,INC    &RO,11,INC    &R1,15,DIS    I,17,INC    A,18,INC    R0,19
185 DATA INC    R1,1A
190 DATA INC    R2,1B,INC    R3,1C,INC    R4/1D/INC    R5,1E,INC    R6,1F,INC    R7
200 REM ***CODES 20...2F***
210 DATA 20,"XCH A/&RO"/21/"XCH A,&R1",25,EN  TCNT I,27,CLR A,28
220 DATA "XCH A,RO",29,"XCH A,R1",2A,"XCH A/R2",2B,"XCH A,R3"
225 DATA 2C,"XCH A,R4"
230 DATA 2D,"XCH A/R5"/2E/"XCH A,R6"/2F/"XCH A/R7",30/"XCHD A,&RO"
240 DATA 31,"XCHD A,&R1",35,DIS TCNT I,37,CPL A,39,"OUTL P1,A",3A
245 DATA "OUTL P2,A"
250 DATA 3C,"MOV P4,A",3D,"MOV P5,A",3E,"MOV P6,A",3F/"MOV P7,A"
260 DATA 40,"ORL A/&RO",41,"ORL A,&R1",42,"MOV A,T",45,STRT CNT,47
265 DATA SWAP A,48
270 DATA "ORL A/RO",49,"ORL A,R1",4A,"ORL A,R2"/4B,"ORL A,R3",4C
280 DATA "ORL A,R4",4D,"ORL A/R5"/4E,"ORL A,R6",4F,"ORL A,R7",50
290 DATA "ANL A,&ROH,51,"ANL A,&R1",55,STRT T,57,DA  A,58
295 DATA "ANL A,RO"
300 DATA 59,"ANL A,R1",5A,"ANL A,R2",5B,"ANL A,R3",5C,"ANL A,R4",5D
310 DATA "ANL A,R5",5E,"ANL A,R6",5F,"ANL A,R7",60,"ADD A,&RO",61
320 DATA "ADD A,&Rf",62,"MOV T,A",65,STOP TCNT,67,RRC A,68,"ADD A,RO"
330 DATA 69,"ADD A,R1",6A,"ADD A,R2"/6B/"ADD A,R3",6C,"ADD A/R4",6D
340 DATA "ADD A/R5"/6E/"ADD A,R6",6F,"ADD A,R7",70,"ADDC A,&RO",71
350 DATA "ADDC A,&R1",E3,"MOVP3 A,&A",75,ENTO CLK,77,RR  A,78
355 DATA "ADDC A,RO",79
360 DATA "ADDC A,R1",7A,"ADDC A/R2",7B/"ADDC A/R3"/7C/"ADDC A,R4",7D
370 DATA "ADDC A,R5",7E,"ADDC A,R6",7F/"ADDC A/R7",80/"MOVX A,&RO",81
380 DATA "MOVX A,&R1",83,RET,85,CLR    FO,8C/"ORLD P4,A",8D/"ORLD P5,A",8E
390 DATA "ORLD P6,A",8F,"ORLD P7,A",90,"MOV    &RO,A",91,"MOV    &R1,A"
395 DATA 93,RETR,95,CPL    FO,97,CLR    C
400 DATA 9C/ANLD P4,A",9D,"ANLD P5,A",9E,"ANLD P6/A"/9F,"ANLD P7,A"
410 DATA AO,"MOV    &RO,A",A1,"MOV    &R1/A"/A3,"MOVP A,&A",A5,CLR    F1
420 DATA A7,CPL C,A8,"MOV RO,A",A9,"MOV R1,A",AA,"MOV R2,A",AB
430 DATA "MOV R3,A",AC,"MOV R4,A",AD,"MOV R5,A",AE
440 DATA "MOV R6,A",AF,"MOV R7,A",B5,CPL F1,C5,SEL RBO,C7
450 DATA "MOV A,PSW",C8,DEC RO,C9,DEC R1,CA,DEC R2,CB,DEC R3,CC
460 DATA DEC R4,CD,DEC R5,CE,DEC R6,CF,DEC R7,DO,"XRL A,&RO",D1
470 DATA "XRL A,&R1",D5,SEL RB1,D7,"MOV PSW,A"/D8,"XRL A,RO",D9
480 DATA "XRL A,R1",DA,"XRL A,R2",DB,"XRL A,R3",DC,"XRL A,R4",DD
490 DATA "XRL A,R5",DE,"XRL A,R6",DF,"XRL A,R7",E5
492 DATA SEL MBO,E7,RL  A,FO
495 DATA "MOV    A,&ROf',Fl,"MOV    A,&R1",F5,SEL    MB1,F7,RCL    A,F8
500 DATA "MOV A,RO",F9,"MOV A,R1",FA,"MOV A,R2",FB,"MOV A,R3",FC
510 DATA "MOV A,R4",FD,"MOV A,R5",FE,"MOV A,R6",FF,IIMOV A,R7"
530 DATA 03,"ADD A,#",13,"ADDC A,#"
540 DATA 16,JTF $,23,"MOV A,#",26,JNTO $,36,JTO $,43,"ORL A,#",46
545 DATA JNT1 $
550 DATA 53,"ANL    A,#ll/56/JT1     $,76,JF1     $,86,JNI    $,88,"ORL    BUS,#",89
560 DATA "ORL P1,#",8A,"ORL P2,#",96,JNZ $,98,"ANL BUS,#",99
565 DATA "ANL P1,#",9A,"ANL P2,#"
570 DATA B0,"MOV &RO,#",B1,"MOV &R1,#",B6,JFO $,B8,"MOV R0,#"
580 DATA B9/MOV R1,#",BA,"MOV R2,#",BB,"MOV R3,#",BC,"MOV R4,#"
590 DATA BD,"MOV R5,#",BE,"MOV Rö^'^BF/MOV R7,#",C6,JZ  $,D3
595 DATA "XRL A,#" E6,JNC $,E8,"DJNZ RO,$"/E9,"DJNZ R1,$"
600 DATA EA,"DJNZ R2,$fl,EB,"DJNZ R3,$",EC,"DJNZ R4,$"
610 DATA ED,"DJNZ R5,$",EE,"DJNZ R6,$",EF,"DJNZ R7,$",F6,JC  $
620 REM DISASSEMBLER-START IST HIER HEX AOOO
630 A=40960
640 INPUT"BILDSCHIRM (1) ODER DRUCKER (4)";X
650 IF X01 AND XO4 THEN 640
670 FOR PA=0 TO 21
680 OP=INT(A/256) AND 15:PRINT H$(OP);:OP=A-INT(A/256)*256
685 GOSUB 130:PRINT OP$;:PRINT "  ";:OP=PEEK(A):GOSUB 130
690 PRINT OP$;" ";:1=0:RESTORE
```

```
700 I=I+1:READ A$
705 IF A$=OP$ THEN PRINT "     ";:READ A$:PRINT A$:GOTO 730
710 IF I>324 THEN 770
720 GOTO 700
730 I=0:A=A+1:RESTORE:NEXT PA
740 IN$=INKEY$:IF IN$="" GOTO 740
750 IF IN$=" " THEN 650
760 END
770 IF (OP AND 31)=4 THEN 880
780 IF (OP AND 31)=20 THEN 910
790 IF (OP AND 31)=18 THEN 940
800 I=I+1 :READ A$:IF A$OOP$ THEN 840
810 GOSUB 980:READ A$:PRINT A$;:IF RIGHT$(A$,1)O"$"THEN 830
820 PRINT H$(C);
830 PRINT OP$:A=A+1:GOTO 730
840 IF I>408 THEN 870
850 GOTO 800
860 END
870 PRINT "== UNBEKANNT==":GOTO 730
880 REM *** JUMP ***
890 B=(OP AND 224)732
900 GOSUB 980:PRINT "JMP $";H$(B);OP$:A=A+1:GOTO 730
910 REM *** CALL ***
920 B=(OP AND 224)732
930 GOSUB 980:PRINT "CALL $";H$(B);OP$:A=A+1:GOTO 730
940 REM *** JUMP ON BIT ***
950 B=(OP AND 224)732
960 GOSUB 980:PRINT "JB";H$(B);" $";H$(C);OP$:A=A+1:GOTO 730
```