

Fri Mar 21 11:17:56 1980

/usr2/leo/cpmvasm

Page 1e

TR CPM SOURCE

*TITLE CPM

•=\$2900 ;STANDARD 16K START ADDRESS

•UNIT•=4 ;WE TELL USER WHAT THE DEFAULT UNIT IS, HERE.
•BDOS•=5 ;VECTOR ADDRESS FOR BDOS CALLS
•FCFB•=\$5C ;WE BUILD AN FCB FOR USER FROM CMD STRING, HERE

;ASCII CHARACTERS

CH•NUL=0 ;NULL FOR ASCIZ
CH•ATN=\$03 ;CONTROL C FOR ATTENTION
CH•TAB=\$09 ;HORIZ TAB CHAR
CH•LF=\$0A ;LINEFEED
CH•CR=\$0D ;CARRIAGE RETURN
CH•XOF=\$13 ;STOP OUTPUT
CH•EOF=\$1A ;END OF FILE (^Z)
CH•CTL=\$20 ;CONTROL CHARS ARE THOSE LESS THAN THIS
CH•SPC=\$20 ;SPACE
CH•NUM=\$23 ;NUMBERSIGN
CH•DOL=\$24 ;DOLLARSIGN FOR STRING TERMINATOR
CH•CLN=\$3A ;COLON
CH•PRM=\$3E ;PROMPT CHARACTER
CH•CM=\$3F ;WILDCARD CHAR, QUESTIONMARK
CH•A=\$41 ;CHARACTER CAPITAL A
CH•UPA=\$5E ;UPARROW FOR TYPING CTL CHARS
CH•LCA=70141 ;LOWER CASE A-Z
CH•LCZ=70172 ;RUBOUT
CH•MSK=\$7F ;MASK FOR 7-BIT CHARACTER

;CORE ADDRESSES

DEFBFR=\$2280 ;DEFAULT DMA ADDR BUFFER
•TPA•=\$30100 ;TRANSIENT PROGRAM AREA
•CCP•=\$2900 ;CONSOLE COMMAND PROCESSOR

;COMMANDS TO BDOS

BF•TYI=1 ;TYPE IN A CHAR
BF•TYO=2 ;TYPE OUT A CHAR
BF•RDR=3 ;GET CHAR FROM READER
BF•PCH=4 ;SEND CHAR TO PUNCH
BF•LPT=5 ;SEND CHAR TO LISTING DEVICE
BF•ADR=6 ;GET ADDR OF BDOS
BF•GST=7 ;GET I/O STATUS BYTE
BF•SST=8 ;SET STATUS BYTE
BF•SOU=9 ;STRING OUTPLT
BF•SIN=10 ;STRING INPUT
BF•KBS=11 ;KEYBOARD STATUS CHECK

;FOLLOWING ARE THE DISK ONES

BF•LHD=12 ;LIFT HEAD
BF•INI=13 ;INIT BDOS, SET DMA TO 80, LOG IN DSK A
BF•LOG=14 ;LOG IN DISK N
BF•OPN=15 ;OPEN FILE

```

BF•CLS=16      ;CLOSE FILE
BF•FND=17      ;FIND FILE
BF•FNX=18      ;FIND NEXT FILE
BF•DEL=19      ;DELETE FILE
BF•RED=20      ;READ NEXT RECORD
BF•WRT=21      ;WRITE NEXT RECORD
BF•CRE=22      ;CREATE NEW FILE
BF•REN=23      ;RENAME FILE
BF•ILV=24      ;INTERROGATE LOCIN VECTOR
BF•ILD=25      ;INTERROGATE LOGGED DRIVE NUMBER
BF•DMA=26      ;SET DMA ADDRESS
BF•IAV=27      ;INTERROGATE ALLOCATION VECTOR FOR CURRENT DISK

```

L

3LAYOUT OF AN FCB (FILE CONTROL BLOCK)

```

FCB•UN=$00      ;UNIT NUMBER, IF FORCED
FCB•NM=$01      ;NAME, 8 CHARS
FCB•EX=$09      ;EXTENSION, 3 CHARS
FCB•RC=$0F      ;RECORD COUNT
FCB•MP=$10      ;MAP AREA
FCB•NR=$20      ;NUMBER OF RECORD

NDISK=4         ;MAX NUMBER OF DISKS ON SYSTEM

XCLUST=$100     ;CLUSTERS ON A DISK, ROUNDED UP
NALLOC=XCLUST/8 ;BYTES OF ALLOC VECTOR FOR A DISK. (8 BITS/BYTE)
NNAMEH=8        ;CHARACTERS IN A FILE NAME
NXTCH=3         ;CHARACTERS IN A FILENAME EXTENSION
NSECH=$80       ;CHARACTERS IN A SECTOR

```

L

```

CCP:   JMP    CCP000      ;MAIN ENTRY TO CCP
      JMP    CCPREE     ;RESTART, CLEARING CMD BUFFER

```

3TEXT BUFFER USED FOR COMMAND TYPEIN.
3 IT IS PRE-LOADED WITH A COPYRIGHT STATEMENT

NCMDBF=7D128 ;SIZE OF BUFFER

```

CROBUF: .BYTE  NCMDBF-1      ;MAXIMUM COUNT OF CHARS IN BUFFER
CBUFCT: .BYTE  2              ;CURRENT COUNT IN BUFFER
CBUFTX: .ASCII  /             /          ;SIXTEEN SPACES
      .ASCII  /COPYRIGHT (C) 1978, DIGITAL RESEARCH /
      .BLKB   NCMDBF+CBUFTX-.}

```

CMDPTR: .ADDR CBUFTX ;PARSER SCAN POINTER

ERRPTR: .BLKW 1 ;POINTER TO TYPE OUT BAD PART IF ERROR

```

CCPTYO: MOV  E, A           ;CHAR TO TYPE
      MVI  C, BF•TYO      ;BDOS TYO FUNCTION
      JMP  .BDOS.          ;CALL BDOS

```

```

TYOSBC: PUSH  B            ;SAME, BUT SAVING BC PAIR
      CALL  CCPTYO      ;CALL TYO ROUTINE
      POP   B            ;RESTORE BC PAIR
      RET

```

TCRLF:	MVI A, CH.CR CALL CCP TYO MVI A, CH.LF JMP CCP TYO	;TYPE CARR RET, LINEFEED ;LINEFEED
CRSCUT:	PUSH B CALL TCRLF POP H	;SAVE POINTER TO ASCIZ ;FIRST A CRLF
CCPSOU:	MOV A, M ORA A RZ INX H PUSH H CALL CCP TYO POP H JMP CCPSOU	;RESTORE POINTER TO ASCIZ ;GET A CHAR FROM STRING ;IS IT A NULL? ;IF SO, END OF STRING ;NOT NULL, STEP PAST IT ;SAVE FOR NEXT LOOP ;TYPE THIS CHAR ;POINT TO NEXT ONE ;LOOP FOR MORE
INIDOS:	MVI C, BF.INI JMP .BDOS.	;FUNCTION = INIT BDOS
LOGDSK:	MOV E, A MVI C, BF.LOG JMP .BDOS.	;THIS IS THE DESIRED DISK ;LOG IN THIS DISK NUMBER
OPNFILE:	MVI C, BF.OPN CALL .BDOS. STA FCBNUM INR A RET	;OPEN A FILE ;SAVE THE BYTE ADDR IN DIR ;IF WAS \$FF, SET Z FLAG ;RETURN Z FLAG TO CALLER
OPNCMD:	XRA A STA CMDFCB+FCB.NR LXI D, CMDFCB JMP OPNFILE	;START OF FILE ;POINT TO FILE BLOCK ;OPEN THE FILE
CLSFIL:	MVI C, BF.CLS CALL .BDOS. STA FCBNUM INR A RET	;CLOSE FILE
FNOFILE:	MVI C, BF.FND CALL .BDOS. STA FCBNUM INR A RET	;FIND FILE
FNDNXT:	MVI C, BF.FNX CALL .BDOS. STA FCBNUM INR A RET	;FIND NEXT FILE IN WILDCARD GROUP ;RETURN Z FLAG IF NO MORE.
FNDCFL:	LXI D, CMDFCB JMP FNOFILE	;FIND FILE
DELFIL:	MVI C, BF.DEL JMP .BDOS.	;DELETE FILE
READFL:	MVI C, BF.RED	;READ NEXT RECORD

CALL •BDOS•
 ORA A
 RET

RECMD: LXI D, CMDFCB ;FCB OF CURRENT COMMAND
 JMP READFL ;READ ANOTHER SECTOR

WRFL: MVI C, BF•WRT ;WRITE NEXT RECORD
 CALL •BDOS•
 ORA A
 RET

CRATE: MVI C, BF•CRE ;CREATE FILE
 CALL •BDOS•
 STA FCBNUM
 RET

RENAME: MVI C, BF•REN ;RENAME FILE
 JMP •BDOS•

CASE: CPI CH•LCA ;RANGE CHECK. LOWER CASE A?
 RC
 CPI CH•LCZ+1 ;RETURN IF SMALLER
 RNC
 ANI \$5F ;LOWER CASE Z?
 RET ;RETURN IF BIGGER
 ;MAKE UPPER CASE

* TO GET A COMMAND, FROM TOP LEVEL COMMAND LOOP *

SELECT: LDA SUBFLG ;SUBMIT FILE IN PROGRESS?
 ORA A
 JZ COLEC2 ;JUMP IF NOT.
 LDA CMDUNI ;LOGGED DISK AT COMMAND LEVEL

MVI A, B ;SELECT "A"
 CNZ LOGDSK ;IF NOT DISK A, SWITCH TO A.
 LDA FC2SUB+FCB•RC ;RECORD COUNT
 DCR A ;LESS ONE

STA FCBSUB+FCB•NR ;FCB OF SUBMIT NAME
 LXI D, FCBSUB ;READ ANOTHER SECTOR
 CALL READFL ;IF ERROR ON READ, STOP.
 JNZ COLEC2 ;MOVE BUFFER TO HERE
 LXI H, CBUFCT ;MOVE BUFFER FROM HERE
 MVI B, NSECCH ;MOVE THIS MANY BYTES
 CALL MOVNCH ;MOVE STRING

; 2A51 21 95 30

LXI H, FCBSUB+FCB•RC. ;FCB FOR SUBMIT NAME
 DCR M ;CLOSE FILE

; 2A54 35

LXI D, FCBSUB ;COMMAND LEVEL DRIVE
 CALL CLSFIL ;SEE IF DRIVE A

JZ COLEC2 ;IF NOT, SWITCH TO WHATEVER IT IS

LDA CMDUNI ;POINT TO THE TEXT

ORA A ;SHOW THE SUB TEXT

CNZ LOGDSK ;ANYTHING TYPED BY OPERATOR?

LXI H, CBUFTX ;IF NOT, USE SUBMIT STUFF

CALL CCPSOU ;IF SO, DELETE SUB FILE.

CALL CHKKBS ;AND GO TO CMD PROMPT

JZ COLEC3

CALL DELSUB

JMP PROMPT

```

COLEC2: CALL    DELSUB      ;DELETE ANY SUB FILE
        MVI C, BF.SIN   ;STRING INPUT FROM TTY
        LXI D, CMDBUF   ;HERE'S WHERE WE WANT TEXT.
        CALL    •BDOS•     ;DO TEXT!
COLEC3: LXI H, CBUFCT   ;SEE HOW MANY CHARS WE GOT
        MOV B, M       ;HERE'S THE COUNT
COLEC4: INX     H       ;LOOP THRU TEXT, RAISING IT.
        MOV A, B       ;COUNT LEFT
        ORA    A       ;ANY LEFT?
        JZ     COLECS   ;QUIT WHEN ALL DONE
        MOV A, M       ;GET A CHAR
        CALL    RAISE   ;IF LOWER CASE ALPHA, RAISE IT
        MOV M, A       ;PUT UPPER CASE IN BUFFER
        DCR    B       ;COUNT THRU BUFFER
        JMP    COLECF4  ;LOOP

COLECF5: MOV M, A      ;MAKE IT ASCIZ - NULL AFTER TEXT
        LXI H, CBUFTX   ;POINT AT START OF TEXT
        SHLD   CMDPTR   ;SCAN STARTS HERE
        RET               ;LINE READY FOR PROCESSING

```

```

CHKKBS: MVI C, BF.KBS   ;CHECK KEYBOARD STATUS
        CALL    •BDOS•
        ORA    A
        RZ
        MVI C, BF.TYI   ;NOTHING TYPED
        CALL    •BDOS•
        ORA    A       ;GET THE TYPED CHAR
        RET               ;MAKE NON-ZERO FLAG (UNLESS NULL)

```

```

LIFTHD: MVI C, BF.LHD   ;LIFT HEAD (NCP)
        JMP    •BDOS•

```

```

GETLDN: MVI C, BF.ILD   ;GET LOGGED DISK NUMBER
        JMP    •BDOS•

```

;UNUSED ROUTINE TO MOVE (C) CHARS FROM (DE) TO (HL)

```

DOBLT: LDAX    D
        MOV M, A
        INX
        INX    H
        DCR    C
        JNZ    DOBLT
        RET

```

```

SETADR: MVI C, BF.DMA   ;SET DMA ADDRESS
        JMP    •BDOS•

```

; 2AC0 C3 05 00

```

DELSUB: LXI H, SUBFLG   ;WAS SUB FILE BEING DONE?
        MOV A, M
        ORA    A
        RZ
        MVI M, B
        XRA    A
        CALL   LOGDSK   ;RETURN IF NOT
        LXI D, FCBSUB  ;IT WAS, BUT NO MORE.

```

;CHOOSE DRIVE A
;LOG IT IN
;SUBMIT FILE'S FCB

Fri Mar 21 11:14:00 1980

YUSREKTEOYCPM&WIM

FIGURE 102

JMP	LOGDSK	J AND RETURN
CHKSER:	LXI D, CCPSER	J MAKE SURE SERIALS MATCH
	LXI H, DOSSER	J BETWEEN CCP AND BDOS
	MVI B, SERSIZ	J CHECK THIS MANY CHARS
CHKSRL:	LDAX D	
	CMP M	
	JNZ SERERR	J IF NOT, GO STEP ON SELF
	INX D	
	INX H	
	DCR B	
	JNZ CHKSRL	J LOOP THRU ALL OF SERIAL
	BFT	J IF MATCHES, RETURN OK

;REJECT COMMAND

CMDREJ:	CALL	TCRLF		;NEW LINE
	LHLD	ERRPTR		;TYPE OUT THE BAD GUY
CMDRJL:	MOV A,	M		; ..
	CPI	CH,SPC		
	JZ	CMDRJ1		;QUIT ON SPACE
	ORA	A		
	JZ	CMDRJ1		;QUIT AT END OF TEXT
	PUSH	H		
	CALL	CCPTYO		;TYPE CHAR OF BAD CMD
	POP	H		
	INX	H		
	JMP	CMDRJL		;LOOP TIL NULL
CMDRJ1:	MVI A,	1?		;STICK "?" ON END
	CALL	CCPTYO		
	CALL	TCRLF		
	CALL	DELSUB		;DELETE SUB FILE, IF ANY
	JMP	PROMPT		;RESTART CCP

L
3 CHECK FOR TERMINATORS
3 RETURN "Z" IF LEGAL SEPARATOR, ABORT ON CONTROLS.

CHKTRM: LDAX D
OR A
RZ ;RETURN Z IF NULL
CPI CH•CTL ;IF CONTROLS,
JC CMDREJ ;ABORT
RZ ;SPACE IS OK, THOUGH
CPI ' = ;SC ARE THE FOLLOWING
RZ
CPI ' _
RZ
CPI ' .
RZ
CPI CH•CLN ;**:
RZ
CPI ' ;
RZ
CPI ' E
RZ
CPI ' J
RZ
RET ;IF NONE OF THOSE RETURN NON-

NOTSPC: LDAX D ;RETURN "Z" ON NULL
 ORA A
 RZ
 CPI \$20 ;SCAN FOR NON-SPACE
 RNZ
 INX D
 JMP NOTSPC

 HL..A: ADD L ;BA + HL => HL
 MOV L, A
 RNC
 INR H
 RET
 -L
 ;ROUTINE TO SCAN A WORD OR FILENAME

 PARSE: MVI A, B ;START AT BEGINNING TO BUILD WORD
 PARSE1: LXI H, CMDFCB ;MAY BE A FILENAME, OR A VERB, ETC
 CALL HL..A ;ADD A TO HL
 PUSH H ;SAVE TWO COPIES
 PUSH H
 XRA A ;ASSUME NOT A DISK SPEC
 STA PRSUNI
 LHLD CMDPTR ;COMMAND POINTER
 XCHG ;TO DE
 CALL NOTSPC ;SCAN FOR NON-SPACE
 XCHG ;BACK TO HL
 SHLD ERRPTR ;IF ERROR, WILL RETYPE FROM HERE
 XCHG ; 2B5D EB
 POP H ; 2B5E E1
 LDAX D ;GET BACK THE NON-SPACE
 ORA A ;WAS NON-SPACE A NULL?
 JZ PARSE2 ;JUMP IF SO
 SBI CH..A-1
 MOV B, A ;MAY BE A DRIVE NAME
 INX D ;SEE IF A COLON IS NEXT
 LDAX D ;COLON?
 CPI CH..CLN ;IF SO, THIS WAS A DRIVE
 JZ PARSE3 ;WASN'T. BACK BEFORE COLON
 DCX D ;PLUG DEFAULT IN

 PARSE2: LDA CMDUNI
 MOV H, A
 JMP PARSE4

 PARSE3: MOV A, B ;WE HAD A <UNIT>:
 STA PRSUNI ;SO PLUG IT IN
 MOV M, B ;
 INX D ;STEP PAST UNIT AND COLON

 PARSE4: MVI B, NNAMCH ;LOOK FOR SPECIFIC TERMINATORS
 PARSE5: CALL CHKTRM ;END OF WORD IF SO
 JZ PARSE9 ;NOT ONE OF THOSE. STEP PAST IT
 INX H

 CPI '*' ;WILDCARD?
 JNZ PARSE6 ;JUMP IF NOT
 MVI H, '?' ;YES, MAKE A WILD CHAR
 JMP PARSE7 ;AND DON'T STEP PAST IT.

 PARSE6: MOV M, A ;STORE CHAR FROM TEXT INPUT

PARSE7: DCR B ;COUNT CHARS IN A NAME
 JNZ PARSES ;SOME LEFT, LOOP.
 PARSE8: CALL CHKTRM ;SKIP TRAILING NON-TERMS
 JZ PARS10 ;QUIT AT TERM
 INX D ;NON-TERM. SKIP IT.
 JMP PARSE8

PARSE9: INX H ;FILL REST OF NAME WITH BLANKS
 MVI M, CH·SPC
 DCR B ;COUNT THRU EIGHT CHARS
 JNZ PARSE9 ;TILL ALL FILLED
 PARS10: MVI B, NEXTCH ;CHARS IN AN EXTENSION
 CPI '.' ;WAS TERM A DOT?
 JNZ PARS15 ;JUMP IF NOT
 INX D ;YES. PASS IT.
 CPI 'x' ;CHECK CHARS IN EXTENSION
 JNZ PARS15 ;GO IF TERMINATOR
 INX H ;STEP DESTINATION
 CPI '?' ;WILD CARD EXTENSION?
 JNZ PARS12 ;JUMP IF NOT
 MVI M, '?' ;YES, MAKE WILD CHAR
 JMP PARS13 ;AND DON'T SCAN PAST IT

PARS12: MOV M, A ;COPY CHAR FROM SOURCE
 INX D ;STEP THRU EXTENSION
 PARS13: DCR B ;COUNT EXTENSION
 JNZ PARS11 ;LOOP TILL DONE
 PARS14: CALL CHKTRM ;SKIP TRAILING NON-TERMS
 JZ PARS16 ;JUMP ON TERM
 INX D ;STEP SOURCE
 JMP PARS14 ;LOOK FOR TERM

PARS15: INX H ;STEP DESTINATION
 MVI M, CH·SPC ;PAD WITH SPACES
 DCR B ;COUNT THRU EXTENSION
 JNZ PARS15 ;FILL WHOLE EXT

PARS16: MVI B, \$03 ;FILL MORE OF SPARE BYTES OF FCB
 PARS17: INX H ;HERE
 MVI M, D ;WITH ZEROS
 DCR B ;COUNT THEM
 JNZ PARS17 ;LOOP FOR THREE SPARE BYTES
 XCHG ;GET HOW MUCH READ FROM TEXT
 SHLD CMDPTR ;REMEMBER FOR NEXT SCAN
 POP H ;BACK TO START OF FILENAME
 LXI B, NNAMCH+NEXTCH ;C GETS # TO SCAN, B GETS ANSWER

PARS18: INX H ;LOOK FOR WILD CARDS IN FILENAME
 MOV A, M ;GET A FILENAME CHAR
 CPI CH·DM ;WILD?
 JNZ PARS19 ;IF NOT, DON'T COUNT IT

INR B ;YES, COUNT IT.
 PARS19: DCR C ;SCAN NAME AND EXT
 JNZ PARS18 ;NUMBER OF WILD CHARS
 MOV A, B ;RETURN Z FLAG AS FAST CHECK
 ORA A
 RET

TL
 CCP COMMAND TABLE

CCCTB: ·ASCII /DIR /
 NEPCH=-CCPCTB

;CHARS IN A COMMAND NAME

•ASCII /ERA/
•ASCII /TYPE/
•ASCII /SAVE/
•ASCII /REN/

△ NCCPC=<.CCPCTB>/NCCPCH

COMMANDS IN TABLE

;SERIAL NUMBER OF COPY OF CCP
;SIZE OF SERIAL NUMBER

HERE TO LOOK UP A COMMAND VERB

FINDCM: LXI H, CCPCTB
MVI C, 0
FINDC1: MOV A, C
CPI NCCPC
RNC
LXI D, CMDNAM
MVI B, NCCPCH
FINDC2: LDAX D
CMP M
JNZ FINDC3
INX D
INX H
DCR B
JNZ FINDC2
LDAX D
CPI CH·SPC
JNZ FINDC4
MOV A, C
RET

;POINT TO LEGAL COMMANDS
;THIS WILL BE INDEX INTO TABLE

;TRIED THEM ALL?
;GIVE UP IF SO
;LOOK AT USER'S STRING
;COMMAND IS THIS LONG

;CHECK A CHARACTER
;NOT THIS COMMAND
;STEP THRU REAL AND TEST STRINGS

;COUNT LENGTH OF COMMAND
;LOOP FOR WHOLE COMMAND
;CHECK USER'S CHAR AFTER COMMAND
;FOR SPACE
;JUMP IF NOT SPACE
;RETURN TABLE INDEX

FINDC3: INX H
DCR B
JNZ FINDC3
FINDC4: INR C
JnP FINDC1

;NOT THIS COMMAND. SKIP REST OF IT.

;TRY NEXT COMMAND

CPNSUB: ORA A
JNZ OPNSB1
LXI D, FCBSUB
CALL CPNFIL
JZ CPNSB1
MVI A, \$FF
JMP OPNSB2

; 2C3A B7
; 2C3B C2 4C 2C
;SUBMIT COMMAND'S FILENAME
;TRY TO FIND IT
;IF NOT THERE, SKIP.
;FLAG DOING SUB FILE

OPNSB1: XRA A
OPNSB2: STA SUBFLG
RET

;NOT DOING SUB FILE
;STORE WHETHER SUBMIT IN PROGRESS OR NOT

CCPREE: XRA A
STA CBUFCT

; 2C51 AF
; 2C52 32 07 29

;JUMP HERE FROM BASE OF CCP

CCP200: LXI SP, CCPSTK
PUSH B
PUSH B
CALL INIDOS
POP B
PUSH PSW

;SET UP STACK FOR CCP

;INIT BDOS

; 2C58 C5
; 2C59 C5
; 2C5D C1
; 2C5E F5

Fri Mar 21 11:20:13 1980

/usr2/leo/cpm.asm

Page 1.9

MOV A, C
CALL LOGDSK
POP PSW
POP B
INR A
ORA C
CALL OPNSUB
LDA CBUFCT
ORA A
JNZ NPROMT
FROMPT: LXI SP, CCPSTK
CALL TCRLF
CALL GETLDN
ADI CH.A
CALL CCPTYO
MVI A, CH.PRM
CALL CCPTYO
CALL COLECT
NPROMT: LXI D, DEFBFR
CALL SETADR
CALL GETLDN
STA CMDUNI
CALL PARSE
CNZ CMDREJ
LDA PRSUNI
ORA A
JNZ ZNOTFD
CALL FINDCM
LXI H, CDSPTB
MOV E, A
MVI D, 0
DAD D
DAD D
MOV A, M
INX H
MOV H, M
MOV L, A
PCHL

; THIS DISK NUMBER
; LOG IT IN AND SELECT IT
; 2C63 F1
; 2C64 C1
; 2C65 3C
; 2C66 B1
; OPEN SUB FILE, IF EXISTS
; 2C6A 3A R7 29
; 2C6D B7
; 2C6E C2 87 2C
; INIT STACK
; 2C74 CD 98 29
; INTERROGATE DRIVE NUMBER
; MAKE IT A LETTER, A-D
; PROMPT CHAR
; GO COLLECT A COMMAND
; DEFAULT LOW CORE BUFFER
; SET DMA ADDRESS
; INTERROGATE DRIVE NUMBER
; SAVE COMMAND LEVEL DISK
; 2C93 CD 44 2B
; 2C96 C4 EF 2A
; 2C99 3A CA 32
; 2C9C B7
; NO SUCH COMMAND
; 2CA0 CD 10 2C
; 2CA3 21 B0 2C
; 2CA6 5F
; 2CA7 16 B0
; 2CA9 19
; 2CAA 19
; 2CAB 7E
; 2CAC 23
; 2CAD 66
; 2CAE 6F
; 2CAF E9

; DISPATCH TABLE FOR TYPED COMMANDS

CDSPTB: •ADDR ZDIR ; DIRECTORY COMMAND
•ADDR ZERA ; ERASE COMMAND
•ADDR ZTYP ; TYPE COMMAND
•ADDR ZSAV ; SAVE COMMAND
•ADDR ZREN ; RENAME COMMAND
•ADDR ZNOTFD ; NO SUCH COMMAND

; HERE ONLY IF SERIAL NUMBER MISMATCHES

SERERR: LXI H, \$76F3 ; MAKE A DI, HLT PAIR
SHLD CCP ; PUT THEM AT START ADDRESS OF CCP
LXI H, CCP ; GO DO THEM
PCHL ; WHEN SERIAL NUMBERS DONT MATCH

RRDERR: LXI B, MRDERR ; TYPE READ ERROR MESSAGE
JMP CRSOUT

MRDERR: •ASCIZ /READ ERROR/

RNTFND: LXI B, MNTFND
JMP CRSOUT

;TYPE NOT FOUND MESSAGE

MNTFND: ASCIZ /NOT FOUND/

GETNUM: LXI H, CMDNAM
LXI B, \$000B
GETNML: MOV A, M
CPI CH·SPC
JZ GETNM1
INX H
SUI '0
CPI '0
JNC CMDREJ
MOV D, A
MOV A, B
ANI # \$E0
JNZ CMDREJ
MOV A, B
RLC
RLC

;POINT AT WORD WHICH MAY BE NUMBER
;11 CHARS, ANS IN B STARTS AT B
;GET WHAT MAY BE A DIGIT
;END OF WORD?
;JUMP IF SO, CHECK TRAILING CHARS
;NO. STEP PAST IT
;SEE IF IT'S A DECIMAL DIGIT
;0-9?
;ERROR IF NOT
;GOOD. KEEP A COPY
;ACCUMULATE DECIMAL NUMBER
;PRE-CHECK FOR OVERFLOW
;BAD IF SO
;OK, MULTIPLY PARTIAL BY 10
;THREE SHIFTS FOR EIGHT

RLC
ADD

B
JC CMDREJ
ADD B
JC CMDREJ
ADD D
JC CMDREJ
MOV B, A
DCR C
JNZ GETNML
RET

;ADD ONE MAKES NINE

;ONE MORE MAKES TEN
;QUIT IF OVERFLOW
;ADD IN NEW DIGIT
;AGAIN, ERR IF OVERFLOW
;NEW PARTIAL ANSWER
;COUNT CHARS IN SOURCE LINE
;LOOP IF MORE TO GO

;MAKE SURE END OF WORD AND NO MORE BUT SPACE FILLS

GETNM1: MOV A, M
CPI CH·SPC
JNZ CMDREJ
INX H
DCR C
JNZ GETNM1
MOV A, B
RET

;ALL SPACES?
;NOT A SPACE
;COUNT AND SKIP ANOTHER
;LOOP IF MORE

~L
MOV3CH: MVI B, \$23
MOVNCH: MOV A, H
STAX D
INX H
INX D
DCR B
JNZ MOVNCH
RET

;MOVE 3 CHARS
;ENTER HERE TO MOVE (B) CHARS
;FROM (HL) TO (DE)

;GET A CHAR FROM DISK DIRECTORY AT (A) + (C) INTO THE BUFFER

GTDRCH: LXI H, DEFBFR
ADD C
CALL HL..A
MOV A, M
RET

;CURRENT CHAR + FCB BASE
;0A + HL => HL
;GET THE CHAR

ROUTINES TO SWITCH TO/FROM DISK NAMED IN A COMMAND, FROM TOP-LEVEL ONE.

TMPSEL:	XRA	A	;CLEAR DISK NUMBER BYTE IN FCB
	STA	CMDFCB	
TMPSEL2:	LDA	PRSUNI	;DISK SPECIFIED?
	ORA	A	
	RZ		;DONE IF NOT
	DCR	A	;YES, CONVERT TO 0-3
	LXI H,	CMDUNI	;WAS IT THE DEFAULT?
	CMP	M	;SAME AS TOP LEVEL?
	RZ		;IF SO, DON'T RE-LOG IT
	JMP	LOGDSK	;DIFFERENT. GO LOG IT.

ROUTINE TO SWITCH TO COMMAND LEVEL DISK.

CFSEL:	LDA	PRSUNI	;WAS THERE ONE IN COMMAND?
	ORA	A	; ••
	RZ		;RETURN IF NOT
	DCR	A	;YES. CONVERT TO 0-3 FORM
	LXI H,	CMDUNI	; 2051 21 C9 30
	CMP	M	;IS IT SAME AS TOP LEVEL ONE?
	RZ		;IF SO, NO PROBLEM
	LDA	CMDUNI	;IF DIFFERENT, SWITCH BACK TO THIS
	JMP	LOGDSK	;GO LOG IT

DIRECTORY LISTING COMMAND. DIR, OR DIR AFN

ZDIR:	CALL	PARSE	;GET THE TEXT ARGUMENT
	CALL	TMPSEL	;SWITCH TO REQUESTED DISK
	LXI H,	CMDNAM	;POINT TO ARGUMENT
	MOV A,	M	;WAS IT BLANK?
	CPI	CH·SPC	
	JNZ	ZDIR2	;NO. DIR OF SPECIFIC FILES
	MVI B,	NNAMCH+NEXTCH	;YES. PLUG IN ALL WILD CARDS
ZDIR1:	MVI M,	CH·QM	;FILL FCB NAME AND EXT
	INX	H	
	DCR	B	;COUNT THEM
	JNZ	ZDIR1	;LOOP FOR WHOLE NAME/EXT
ZDIR2:	CALL	FNDCL	;FIND FILE
	CZ	RNTEND	;IF NONE AT ALL, TYPE "NOT FOUND"
ZDIR3:	JZ	ZDIR7	;JUMP IF NONE FOUND
	CALL	TCRLF	;NEXT LINE
	CALL	GETLN	;FIND LOGGED DISK NUMBER
	ADI	CH·A	;MAKE "A"-"D"
	CALL	CCPTYO	;DISK NAME
	MVI A,	CH·CLN	
	CALL	CCPTYO	;COLON
	MVI A,	CH·SPC	
	CALL	CCPTYO	;SPACE FOR SEPARATOR
	LDA	FCBNUM	;COMPUTE BYTE ADDR IN SECTOR
	RAL		;FOR THIS FCB, FROM FCB NUMBER
	RAL		
FAI:	\$60		;32 BYTES PER FCB, FOUR FCB PER SECTOR
	MOV C,	A	;HOLD HERE
	MVI B,	\$01	;FIRST CHAR OF FILENAME
ZDIR4:	MOV A,	B	;CHARACTER NUMBER
	CALL	GTDRCN	;GET DEF BFR+C+A TO A
	CPI	CH·SPC	;TRAILING SPACE IN NAME?

```

        JNZ      ZDIR5          ;JUMP IF NOT
        MVI A, $09           ;IF SO, SKIP TO EXTENSION
        CALL    GTDRCH         ;GET EXTENSION'S FIRST CHAR
        MVI A, CH.SPC         ;IS EXT BLANK?
        CMP     M              ; ..
        JZ      ZDIR6          ;JUMP IF SO.
        CALL    TYOSBC         ;TYO, SAVING BC
        INR     B              ;STEP TO NEXT CHAR OF NAME/EXT

ZDIR5:   CALL    TYOSBC
        INR     B
        MOV A, B
        CPI     $EC
        JNC    ZDIR6
        CPI     $09
        JNZ    ZDIR4
        MVI A, CH.SPC
        CALL    TYOSBC
        JMP     ZDIR4

;L
ZDIR6:   CALL    CHKKBS
        JNZ    ZDIR7
        CALL    FNDNXT
        JMP     ZDIR3

ZDIR7:   JMP     CMDDUN          ;ON TYPEIN, DONE WITH COMMAND

```

;ERASE FILE(S) COMMAND

```

ZERA:    CALL    PARSE          ;SCAN NAME(S)
        CPI     NNAMCH+NEXTCH
        JNZ    ZERA1          ;ALL FILES?
        LXI B, ERAMSG         ;NOT ALL WILDCARDS
        CALL    CRSOUT         ;YES, BETTER CHECK THAT!
        CALL    COLECT         ;ASK OPERATOR
        LXI H, CBUFCT         ;GET A Y OR N
        DCR    M              ;HOW MANY CHARS IN ANSWER
        JNZ    PROMPT         ;BETTER HAVE BEEN ONE
        PROMPT
        INX    H              ;JUMP IF WASN'T ONE.
        MOV A, M              ;WAS ONE CHAR. GET IT.
        CPI     'Y
        JNZ    PROMPT         ;WAS IT A YES?
        QUIT IF NOT.
        INX    H              ;REALLY WANTS ALL FILES DELETED
        SHLD   CMOPTR         ;POINT AT THE TRAILING SPACES
        LXI H, CMDFCB         ;AND RE-BUILD THE ".*."
        MVI M, CH.QM          ;OUT OF ALL WILD-CARDS
        CALL    TMPSL2         ;SWITCH TO CORRECT DISK
        LXI D, CMDFCB         ;POINT TO THE WILD NAME
        CALL    DELFIL         ;DO THE DELETES
        CALL    INIDOS         ;RE-INIT THE DOS
        JMP     CMDDUN         ;SCAN TO EOL AND RE-PROMPT

ZERA1:   CALL    TMPSEL         ;SWITCH TO REQUESTED DISK
        LXI D, CMDFCB         ;POINT TO REQUESTED FILE NAME
        CALL    DELFIL         ;DO THE DELETES
        JMP     CMDDUN         ;SCAN TO EOL AND RE-PROMPT

```

ERAMSG: •ASCIZ •ALL FILES (Y/N)? ;FOR VERIFYING *.*

;COMMAND TO TYPE OUT A FILE

ZTYP: CALL PARSE ;GET THE FILE NAME TO TYPE
 INZ CMOREJ ;I CAN'T DO WILDCARDS

Fri Mar 21 11:21:15 1980

/usr2/leo/cpm.asm

Page 1•13

CALL	TMPSEL	;SWITCH TO REQUESTED DISK	
CALL	OPNCMD	;OPEN THE FILE	
JZ	ZTYPF	;JUMP IF NOT FOUND	
CALL	TCRLF	;MOVE BELOW COMMAND LINE	
LXI H,	TYPNCH	;CHAR COUNT IN SECTOR (USED CT)	
MVI M,	SFF	;MAKE HUGE	
TYP1:	LXI H,	TYPNCH	;SHOW MANY CHARS USED
MOV A,	M	;READ THIS MANY?	
CPI	NSECCH	;WHOLE SECTOR?	
JC	ZTYP2	;JUMP IF STILL OK	
PUSH	H	;NEED TO READ A SECTOR. SAVE HL	
CALL	REDCMD	;READ NEXT SECTOR	
POP	H	;RESTORE STACK / HL	
JNZ	TYEOFQ	;JUMP IF EOF OR ERR.	
XRA	A	;CLEAR COUNT OF TYPED CHARS IN SECTOR	
MOV M,	A	;•••	
TYP2:	INR	M ;COUNT A USED CHAR	
LXI H,	DEFBFR	;POINT INTO BUFFER	
CALL	HL••A	;ADD IN USED COUNT	
MOV A,	M	;GET THE CHAR AT THAT POINT	
CPI	CH•EOF	;END OF FILE?	
JZ	CHDDUN	;NO, TYPE THE CHAR.	
CALL	CCPTYC	;CHECK THE KEYBOARD	
CALL	CHKKBS	;IF TYPEIN, QUIT.	
JHZ	CHDDUN	;LOOP, TYPE MORE.	
JMP	ZTYP1		
MEFC0:	DCR	A ;WAS IT ERR OR EOF?	
JZ	CHDDUN	;IF EOF, THAT'S ALL.	
CALL	RRDERR	;ERROR. SAY "READ ERROR"	
TYPF:	CALL	DFLSEL ;SWITCH BACK TO DEFAULT UNIT	
JMP	CMDREJ	;AND ERROR PATH BACK TO IGP	

SAVE COMMAND

ZSAV:	CALL	PARSE	;SCAN A SIZE NUMBER
	LDA	PRSPUNI	;DISK SPECIFIED IN SIZE??
	ORA	A	;•••
	JNZ	CMDREJ	;IF SO, THAT'S WRONG.
	CALL	GETNUM	;GET SIZE FROM COMMAND TEXT
	PUSH	PSW	;SAVE THE COUNT
	CALL	PARSE	;NOW GET THE FILENAME
	JNZ	CMDREJ	;NO WILDCARDS ALLOWED
	CALL	TMPSEL	;SWITCH TO REQUESTED DISK
	LXI D,	CMDFCB	;DELETE ANY PREVIOUS COPY
	PUSH	D	;SAVE ADDR OF CMDFCB
	CALL	DELFILE	;DELETE
	POP	D	;POINT AT NAME AGAIN
	CALL	CREATE	;CREATE NEW COPY
	CALL	OPNCMD	;OPEN NEW COPY
	JZ	ZSAV2	;JUMP IF NO SPACE.
	POP	PSW	;NUMBER OF PAGES TO SAVE
	MOV L,	A	;CONVERT TO SECTORS
	MVI H,	B	;WHICH MAY BE 9 BITS
	DAD	H	;BY DOUBLING (NOT PARAMETERIZABLE?)
	LXI D,	•TPA•	;START AT BASE OF TRANSIENT AREA
ZSAV1:	MOV A,	H	;SEE IF DONE ALL SECTORS
	ORA	L	;•••
	JZ	ZSAV3	;IF ZERO, DONE.
	DCX	H	;COUNT ANOTHER SECTOR SAVED

PUSH	H	;SAVE COUNT
LXI H,	NSECCH	;ADD ONE SECTOR
DAD	D	;TO CURRENT PLACE IN CORE
PUSH	H	;SAVE WHERE TO DUMP FROM
CALL	SETADR	;SET DMA ADDRESS TO THERE
LXI D,	CMDFCB	;FILE WE ARE WRITING
CALL	WRITFL	;WRITE ANOTHER RECORD
POP	D	;RESTORE CURRENT ADDRESS
POP	H	;AND COUNT OF SECTORS LEFT TO DO
JNZ	ZSAV2	;JUMP IF ERROR ON WRITE
JMP	ZSAV1	;ELSE GO SAVE SOME MORE
ZSAV2:	LXI B,	ZSAVM1 ;NO SPACE ERROR MESSAGE
	CALL	CRSOUT ;TYPE IT
ZSAV3:	LXI D,	CMDFCB ;POINT AT FILE
	CALL	CLSFILE ;CLOSE IT, DO WRITE ALLOC BYTES
	JNZ	ZSAV4 ;JUMP IF OK
	LXI B,	ZSAVM2 ;ERROR ON CLOSE.
	CALL	CRSOUT ;SAY SO
ZSAV4:	JMP	CMDUN ;END OF SAVE COMMAND
ZSAVM1:	ASCIZ /NO SPACE/	
ZSAVM2:	ASCIZ /CANNOT CLOSE/	
 L		
BRENAME COMMAND - REN NEW=OLD		
ZREN:	CALL	PARSE ;GET NEW FILE NAME
	JNZ	CMDREJ ;ERROR IF WILDCARDS
	LDA	PRSUNI ;ANY DISK?
	PUSH	PSW ;SAVE FOR LATER CHECK
	CALL	TPPSEL ;SWITCH TO REQUESTED DISK
	CALL	FNDCL ;FIND NEW FILE
	JNZ	ZREN5 ;JUMP IF FOUND (SHOULDN'T)
	LXI H,	CMDFCB ;COPY OUT THE FILE NAME
	LXI D,	CMFCB2 ;TO HERE
	MVI B,	\$10 ;THIS MANY CHARACTERS
	CALL	MOVNCH ;MOVE THAT TEXT
	LHLD	CHDPTR ;RESTART SCAN
	XCHG	
CALL	NOTSPC	;SCAN FOR A NON-SPACE
	CPI	'=' ;LEGAL SEPARATORS
	JZ	ZREN1 ;OK
	CPI	'_ ;OTHER LEGAL SEPARATOR
	JNZ	ZREN4 ;ERROR IF NOT
ZREN1:	XCHG	
INX	H	;STEP PAST THE "=="
	SHLD	CMDPTR ;RE-SCAN FROM THIS POINT
	CALL	PARSE ;GET OLD FILE NAME
	JNZ	ZREN4 ;NO STARS ALLOWED
	POP	PSW ;NEW DISK NUMBER
	MOV B,	A ;COMPARE WITH OLD FILE'S DISK
	LXI H,	PRSUNI ;DISK IN SECOND NAME
	MOV A,	M ;WAS THERE ONE?
	ORA	A ;..
	JZ	ZREN2 ;JUMP IF NOT
	CMP	B ;YES. SAME AS OLD FILE?
	MOV M,	B ;AND UPDATE IT
	JNZ	ZREN4 ;CAN'T RENAME ACROSS DISKS
ZREN4:	MOV M,	D ;THIS IS THE ONE.

Fri Mar 21 11:21:48 1980

/USRZ/1EO/CDM.ASM

Page 103

XRA A ;CLEAR DISK BYTE IN NAME
STA CMDFCB ;
CALL FNDCFL ;FIND OLD FILE.
JZ ZREN3 ;IF NOT FOUND, QUIT
LXI D, CMDFCB ;DO THE RENAME!
CALL RENAMF ;DONE
JMP CMDDUN ;

ZREN1: CALL RNTEND ;OLD FILE NOT FOUND MSG
JMP CMDDUN ;

ZREN2: CALL DFLSEL ;SWITCH BACK TO DEFAULT UNIT
JMP CMDRDJ ;

ZREN3: LXI B, FEXMSG ;NEW FILE ALREADY EXISTS MSG
CALL CRSOUT ;AND NO MORE TO DO
JMP CMDDUN ;

FEING: .ASCIZ /FILE EXISTS/
TL
ZNCFD: CALL CHKSER ;NO SUCH CMD. IS VERSION OF SYSTEM OK?
LDA CHDNAM ;YES. CMD START WITH A SPACE?
CPI CH.SPC ;
JNZ FCMDQ ;IF NOT BLANK COMMAND, JUMP
LDA PRSUNI ;DISK DRIVE ALONE?
ORA A ;
JZ CMDDN1 ;IF NOT, PROBABLY BLANK LINE
DCR A ;THERE WAS A DRIVE. CONVERT TO 0-3
STA CHDUNI ;KEEP A COPY
STA •UNIT• ;AND ONE FOR THE USER
CALL LOGDSK ;GO LOG THAT DISK
JMP CMDDN1 ;CHECK FOR JUNK ON END OF LINE

RCCS: LXI D, CMDFCB+FCB•.EX ;FIRST CHAR AFTER 8-LTR NAME
LDAX D ;
CPI CH.SPC ;SPACE?
JNZ CMDRDJ ;IF NOT, ERROR.
RCCS1: PUSH D ;SAVE POINTER TO EXTENSION
CALL TMPSEL ;SWITCH TO REQUESTED DISK
POP D ;EXTENSION AGAIN
LXI H, COMEXT ;PLUG IN .COM EXTENSION
CALL MOV3CH ;COPY IT INTO FCB
CALL OPNCMD ;TRY TO OPEN SUCH A FILE
JZ RCOME ;JUMP IF FAILED. RE-LOG CMD DISK
LXI H, •TPA• ;BASE OF TRANSIENT AREA
PUSH H ;SAVE CORE ADDR
XCHG ;IN DE FOR SUBR ARG
CALL SETADR ;SET DMA ADDRESS
LXI D, CMDFCB ;THIS IS THE FILE
CALL READFL ;READ ANOTHER SECTOR
JNZ RCOM2 ;IF EOF OR ERROR.
POP H ;ADDRESS AGAIN
LXI D, NSECCH ;SIZE OF SECTOR
DAD D ;MOVE UP THAT MUCH
LXI D, CCP ;BASE OF CCP
MOV A, L ;MAKE SURE NOT OVERWRITING SELF
SUB E ;16 BIT SUBTRACT
MOV A, H ;
SBB D ;
INC RCOMY ;JUMP IF TOO BIG

	JMP	RCOM1	ELSE READ SOME MORE
RCOM2:	POP	H	;CLEAR ADDR FROM STACK
	DCR	A	;BETTER BE EOF
	JNZ	RCOMX	;IF NOT, ERRCR
	CALL	DFLSEL	;SWITCH BACK TO DEFAULT UNIT
	CALL	PARSE	;SEE IF ANY TRAILING ARGS
	LXI H,	PRSUNI	;GET DISK DRIVE NUMBER
	PUSH	H	;SAVE A COPY OF ADDR
	MCV A,	M	;PICK UP DRIVE
	STA	CMDFCB	;PUT A COPY IN FCB OF FILE
	MVI A,	\$10	;MOVE DOWN IN THE FCB THIS FAR
	CALL	PARSE1	;AND PARSE ANY ARGUMENTS
	POP	H	;DRIVE ADDRESS AGAIN
	MOV A,	M	;MAY HAVE BEEN A DRIVE IN THE ARG
	STA	CMFCB2	
	XRA	A	;START OF FILE
	STA	CMDFCB+FC5.NR	
	LXI D,	*FCFB.	;COMMAND FCB IN LOW CORE
	LXI H,	CMDFCB	;POSSIBLE ARG OF FILENAME
	MVI B,	\$21	;THIS MANY CHARS FOR USER
	CALL	MOVNCH	;COPY THEM
	LXI H,	CBUFTX	;RE-SCAN TO GIVE USER TEXT
RCOM3:	MOV A,	M	;LOOK FOR SPACE OR NULL
	CRA	A	;;;;
	JZ	RCOM4	;QUIT ON NULL
	CPI	CH _n SPC	;QUIT ON SPACE
	JZ	RCOM4	;;;;
	INX	H	;KEEP LOOKING
	JMP	RCOM3	
RCOM4:	MVI B,	B	
	LXI D,	DEFBFR+1	;COUNT
RCOM5:	MOV A,	M	;COPY TRAILING STUFF FOR USER
	STAX	D	;FROM COMMAND STRING
	CRA	A	;TO USER COMM REGION
	JZ	RCOM6	;QUIT ON NULL
	INR	B	;;;;
	INX	H	;COUNT
	INX	D	;SOURCE
	JMP	RCOM5	;DEST
			;AND LOOP
RCOM6:	MOV A,	B	
	STA	DEFBFR	;SHOW MANY CHARS WE COPIED
	CALL	TCRLF	;TELL USER. (HE GETS NO NULL)
	LDA	CMDUNI	;BEFORE GOING TO USER, CRLF
	STA	*UNIT.	;TELL USER THE CURRENT DRIVE
	CALL	LIFTHD	;;;;
	LXI D,	DEFBFR	;LIFT HEAD (CCP) TO BDOS
	CALL	SETADR	;PUT DMA BACK TO DEFAULT
	CALL	*TPA.	;SET DMA ADDRESS
	LXI SP,	CCPSTK	;CALL THE USER PROGRAM
	LDA	CMDUNI	;IN CASE IT STEPPED ON STACK
	CALL	LOGDSK	;IN CASE IT CHANGED DISKS
	JMP	PROMPT	;RE-LOG THE DEFAULT
			;AND GO GET ANOTHER COMMAND
RCOME:	CALL	DFLSEL	
	JMP	CMDREJ	;SWITCH BACK TO DEFAULT UNIT
			;ERROR RETURN
RCOMX:	LXI B,	LDERRM	;LOAD ERROR COMMENT

CALL CRSOUT
JMP CMDDUN

;ASK FOR ANOTHER COMMAND

LDETRM: •ASCIIZ /LOAD ERROR/

COMKT: •ASCII /COM/

CMDUN: CALL DFLSEL ;SWITCH BACK TO DEFAULT UNIT
CMDINI: CALL PARSE ;ANY MORE TEXT?
LDA CHDNAM ;SHOULD BE JUST BLANKS
SUI CH•SPC ;
LXI H, PRSUN1 ;MAYBE A DISK GOT PARSED
ORA M ;
JNZ CMDREJ ;IF EITHER, ERROR.
JMP PROMPT ;OK, GET ANOTHER COMMAND

•BLKB \$10

CCPTK:

SUBFLG: •BYTE 0

FCTSUB: •BYTE 2
•ASCII /\$\$\$ SUB/
•BLKB 21

;CCP'S STACK ABOVE HERE
;FLAG FOR SUBMIT FILE BEING PROCESSED

CFTSUB: •BLKB 1
CHCMDM: •BLKB 15
CMTR32: •BLKB \$11

;FCB STARTS HERE W/ DRIVE NUMBER
;USER'S COMMAND VERB OR FILENAME GOES HERE
;ANOTHER ONE FOR RENAMES

FCCJUM: •BLKB 1
CHCMDI: •BLKB 1
PRTRNI: •BLKB 1
TYPTCH: •BLKB 1

;UNIT AT COMMAND LEVEL
;UNIT RECOGNIZED BY PARSER
;TEMP FOR TYPE COMMAND

;MOVE TO PAGE BOUNDARY

•BLKB 52

;***** END OF CONSOLE COMMAND PROCESSOR *****

;***** BDOS STARTS HERE *****

DOSSER: •BYTE 0, \$0E, 0, 0, 1, 2 ;SERIAL NUMBER OF COPY OF DOS
SET 17 = -DOSSER ;SIZE OF SERIAL NUMBER

;ENTER HERE FOR ALL SERVICES. 5/ JMP BDOS

BDOSH: JMP BDOSH ;THIS IS A VECTOR. GO TO BODY.

VERISC: •ADDR BADSEC ;HANDLER FOR BAD SECTOR
VSSELR: •ADDR SELERR ;HANDLER FOR SELECT ERROR
VFUTRR: •ADDR ROERR ;HANDLER FOR R/O ERROR

;FOLLOWING ROUTINE AND CONSTANTS ARE MEDIUM-DEPENDANT

;ROUTINE TO PICK UP SKEWED SECTOR NUMBER

SKEISC: LXI H, SKEWTB ;POINT TO TRANSLATION TABLE
MVI B, B ;MAKE SECTOR INTO FULL WORD
DAD B ;INDEX INTO TABLE
MOV C, M ;PICK UP TRANSLATED SECTOR
JMP CBSTSC ;BIOS VECTOR TO SET SECTOR

;THIS IS THE ONLY SUCH CALL)

•BYTE 0

;TABLE OF SECTORS, LENGTH 26. FOR SKEW

SKEWTB: •BYTE \$01,\$07,\$0D,\$13,\$19
 •BYTE \$05,\$0B,\$11,\$17
 •BYTE \$03,\$09,\$0F,\$15
 •BYTE \$02,\$08,\$2E,\$14,\$1A
 •BYTE \$06,\$0C,\$12,\$18
 •BYTE \$04,\$0A,\$10,\$10
 •BYTE 0,0,0,0,0,0

;CONSTANTS (READ ONLY) DEFINING DISK LAYOUT

SECPTK: •BYTE \$1A ;SECTORS PER TRACK
 MAXFC5: •BYTE \$3F ;MAX FCB IN DISK DIRECTORY
 LSECPC: •BYTE 3 ;LOG2 OF CLUSTER SIZE. 8 SECTORS PER ALLOC UNIT
 ALLMSK: •BYTE 7 ;MASK FOR SECTOR WITHIN ALLOC UNIT
 MACLUS: •BYTE \$F2 ;N CLUSTERS FOR ALLOCATOR
 INIALO: •BYTE \$C0 ;INITIAL ALLOC. ALWAYS PRESERVE FIRST 2 CLUSTERS,
 ; WHICH ARE THE DIRECTORY
 TKBIAS: •BYTE \$C2 ;EIAS ON ALL COMPUTED TRACKS. SKIPS BOOT IMAGE.

TL
 ;HERE TO HANDLE CALLS THRU 5.

BDOSH:	XCHG	;USER'S INFO ADDR TO HL
	SHLD	USERDE ;SAVE IT HERE
	LXI H,	USRFCN ;PUT USER'S FUNCTION HERE
	MOV M,	C ;..
	LXI H,	D ;SAVE USER'S STACK POINTER
	DAD	SP ;WHICH TAKES A BIT OF EFFORT
	SHLD	USERSP ;STASH IT HERE
	LXI SP,	BDOSSP ;LOCAL STACK
	CALL	BDOSB ;CALL GUTS AS A SUBR
	LHLD	USERSP ;RESTORE USER STACK
	SPHL	;..
	LHLD	ANSWER ;GET ANSWER TO USER
	MOV A,	L ;IN A AND B FOR PL/M CODE
	MOV B,	H
	RET	

BADSEC:	LXI H,	M\$ADSC ;BAD SECTOR MESSAGE
	CALL	ERRPRT ;ERROR MSG PRINTER
	CPI	CH•ATN ;USER TYPED "C?"
	JZ	\$00B0 ;IF SO, REBOOT
	RET	;IF NOT, ACCEPT BAD DATA

SFLERR:	LXI H,	M\$ELEC ;'SELECT' MESSAGE
	JMP	ERROR1 ;BIGGER ERROR MSG. AND REBOOT

ROERR:	LXI H,	M\$ONLY ;'R/O' MESSAGE
ERROR1:	CALL	ERRPRT ;TYPE BIGGER ERR MSG
	JMP	\$00B0 ;RE-BOOT AS IF "C"

ERRPRT:	PUSH	H ;SAVE SPECIFIC MSG
	CALL	BDCRLF ;TYPE A CRLF
	LDA	CURDSK ;GET DRIVE NUMBER
	ADI	CH•A ;MAKE IT A LETTER. A-D

STA	MSG002	;STUFF IT INTO TEXT OF MSG
LXI B,	MSG001	;GET START OF MSG
CALL	DLSOUT	;TYPE THE GENERAL MSG
POP	B	;GET SPECIFIC MSG
CALL	DLSOUT	;TYPE IT, TOO
JMP	TYISVQ	;GOBBLE A CHARACTER AND RETURN

;L
;TEXT FOR ABOVE MESSAGE PRINTERS

MSG001: •ASCII /BDOS ERR ON /
 MSG002: •ASCII / : \$/
 MBADSC: •ASCII /BAD SECTOR\$/
 MSFLEC: •ASCII /SELECT\$/
 MROMLY: •ASCII •R/D\$.

;ROUTINE TO READ A CHAR FROM TYI. FIRST SEE IF ONE SAVED ALREADY.

TYISVQ:	LXI H, SAVCH	;SEE IF THERE WAS A SAVED CH,
	MOV A, M	;FROM STOPPING TYPEOUT
	MVI H, D	;SEE IT ONLY ONCE
	ORA A	;WAS THERE ONE?
	RNZ	;RETURN IT IF SO
	JMP CBCTYI	;ELSE GO GET ONE FROM BIOS

;GET AND ECHO A CHARACTER

RDEONS:	CALL TYISVQ	;GET SAVED, OR NEW, CHAR
	CALL CKPRNT	;CHECK IT FOR CONTROLS
	RC	;RETURN IF NON-TYPING CTL
	PUSH PSW	;ECHO IT.
	MOV C, A	;DIFFERENT AC FOR TYO
	CALL CTY02	;TYPE IT OUT
	POP PSW	
	RET	

;L
;CHECK FOR PRINTABLE CHAR. SIMPLE CONTROLS ARE PRINTABLE.
;RETURN WITH CY FLAG ON IF ITS A NON-PRINTABLE CTL CHAR, NEEDING " "

CKPRNT:	CPI CH•CR	
	RZ	
CPI	CH•LF	
	RZ	
CPI	CH•TAB	
	RZ	
CPI	CH•CTL	
	RET	

CHKKBD:	LDA SAVCH	;GET ANY SAVED CHAR
	ORA A	
	JNZ CHKKB2	;RETURN IT
	CALL CBCTCK	;CHECK KEYBOARD STAT, IN BIOS
	ANI \$81	
	RZ	;RETURN IF NONE THERE
	CALL CBCTYI	;THERE'S SOMETHING READY. GET IT.
CPI	CH•XOF	
JNZ	CHKKB1	;JUMP UNLESS STOP REQUEST
CALL	CBCTYI	;IN WHICH CASE WAIT FOR ANOTHER CH ← HA!
CPI	CH•ATN	;IS IT A CONTROL C?
JZ	\$0000	;IF SO, RE-BOOT
XRA	A ←	;AFTER RESUMING, RETURN ↵

RET

CHKKB1: STA SAVCH
 CHKKB2: MVI A, \$01
 RET

;SOMETHING BUT NOT XOFF. SAVE IT.
 ;SAY HAVE SOMETHING TO READ

~L
 ;CONSOLE TYO ROUTINE

CTYO: PUSH B
 CALL CHKKBD
 POP B
 PUSH B
 CALL CBCTYO
 POP B
 PUSH B
 LDA LPTELG
 ORA A
 CNZ CBLPTO
 POP B
 MOV A, C
 LXI H, HPOS
 CPI CH.RUB
 RZ
 INR M
 CPI CH.CTL
 RNC
 DCR M
 CPI CH.LF
 RNZ

;SAVE CHAR IN C
 ;CHECK FOR BEING FROZEN
 ;RESTORE AND SAVE CHAR AGAIN
 ;TTY OUTPUT IN BIOS
 ;GET BACK CHAR AND SAVE AGAIN
 ;COPYING ON PRINTER?
 ;IF SO, PRINT IT.
 ;GET THE CHAR AGAIN
 ;COPY IN A
 ;POINT TO COLUMN POSITION
 ;A RUBOUT DOESN'T SPACE
 ;SO DON'T. DONE.
 ;STEP HPOS
 ;CONTROL CHAR?
 ;IF NOT, ALL DONE.
 ;YES. DON'T SPACE AFTER ALL
 ;ON LF (SHOULD BE CR),

MVI M, B
 RET

~L
 ;HERE FOR PRETTY TYO, WITH "A FORMAT AND TAB SIMULATION

CTY01: MOV A, C
 CALL CKPRNT
 JNC CTY02
 PUSH PSW
 MVI C, CH.UPA
 CALL CTYO
 POP PSW
 ORI \$40
 MOV C, A
 CTY02: MOV A, C
 CPI CH.TAB
 JNZ CTY0
 CTY0TL: MVI C, CH.SPC
 CALL CTYO
 LDA HPOS
 ANI \$07
 JNZ CTY0TL
 RET

;COPY CHAR
 ;CHECK FOR NEEDING UPARROW
 ;JUMP IF DON'T.
 ;SAVE CHAR IN A
 ;GET AN UPARROW
 ;TYPE UPARROW
 ;GET BACK CHAR
 ;MAKE UN-CONTROL.
 ;BACK TO RIGHT AC
 ;COMMON TYO ROUTINE
 ;TAB CHAR?
 ;IF NOT, TYPE IT AS IS.
 ;SIMULATE TAB WITH SPACES
 ;TYPE A SPACE
 ;ENOUGH TO MAKE MULT OF 8?
 ;IF NOT, TYPE SOME MORE.

NMCRLF: MVI C, CH.NUM
 CALL CTYO
 BDCRLF: MVI C, CH.CR
 CALL CTYO
 MVI C, CH.LF
 JMP CTYO

;NUMBER SIGN
 ;TYPE THE NUMBERSIGN
 ;TYPE CR
 ;AND LINEFEED
 ;TYPE AND RETURN

;TYPE STRING UP TO TERMINATING DOLLARSIGN

DLSOUT:	LDAX B	;GET CURRENT CHAR
	CPI CH.DOL	;IS IT A DOLLAR?
	RZ	;IF SO, RETURN WITHOUT PRINTING IT
	INX B	;ELSE, STEP FOR NEXT ONE,
	PUSH B	;SAVE POINTER,
	MOV C, A	;MOVE CURRENT CHAR TO C,
	CALL CTY02	;AND TYPE CURRENT CHAR
	POP B	;GET BACK MEMORY POINTER
	JMP DLSOUT	;GO DO ANOTHER CHAR

;HERE FOR READ STRING BUFFERED (PSIN) COMMAND

PSIN:	LHLD USERDE	; 326B 2A 0A 33
	MOV C, M	; 326E 4E
	INX H	; 326F 23
	PUSH H	; 3270 E5
	MVI B, 0	; 3271 66 00
PSIN1:	PUSH B	;SAVE SOME REGS
	PUSH H	
PSIN2:	CALL TYISVO	;RAW TYI, OR SAVED ONE FROM TYC/STOP
	ANI CH.MSK	;JUST 7 BITS
	POP H	
	POP B	
	CPI CH.CR	;HANDLE CAR RET
	JZ PSIN13	;TERMINATE STRING
	CPI CH.RUB	;RUBOUT?
	JNZ PSIN3	;JUMP IF NOT
	MOV A, B	;HANDLE CHAR DELETION
	ORA A	
	JZ PSIN1	
	MOV A, M	
	DCR B	
	DCX H	
	JMP PSIN11	
		; 3287 B7
		; 3288 CA 73 32
		; 328B 7E
		; 328C 65
		; 328D 2B
		; 328E C3 E6 32
PSIN3:	CPI \$05	;CONTROL E? (PHYS CRLF)
	JNZ PSIN4	
	PUSH B	
	PUSH H	
	CALL BDCRLF	
	JMP PSIN2	
		; 3293 C2 9E 32
		; 3290 C5
		; 3297 E5
		; 3298 CD 53 32
		; 329B C3 75 32
PSIN4:	CPI \$10	;CONTROL P?
	JNZ PSIN5	;JUMP IF NOT
	PUSH H	;SAVE STRING POINTER
	LXI H, LPTFLG	;TOGGLE PRINTER-COPY FLAG
	MVI A, \$01	; • •
	SUB M	; • •
	MOV M, A	; • •
	POP H	;RESTORE POINTER
	JMP PSIN1	;GET ANOTHER CHAR.
PSIN5:	CPI \$18	;CONTROL X?
	JZ PSIN6	
	CPI \$15	
	JNZ PSIN7	
PSIN6:	CALL NMCRLF	;HERE TO DELETE WHOLE LINE, TYPE "##"
	POP H	; AND A CRLF
		; 32B1 CA B9 32
		; 32B6 C2 C0 32

תשרי ט' 1980 11:25:50 Mar 21 Fri

Justitia secessit

Page 146

JMP	PSIN	TRY AGAIN
PSIN7:	CPI \$12	;CONTROL R?
JNZ	PSIN10	;JUMP IF NOT
PUSH	B	;YES. RETYPE LINE
CALL	NMCRLF	;TYPE "##" CRLF FIRST
POP	B	;RESTORE COUNT
POP	H	;AND POINTER
PUSH	H	;THEN RE-SAVE
PUSH	B	;AND COUNT
PSIN8:	MOV A, B	;CHECK COUNT FOR DONE
ORA	A	;..
JZ	PSIN9	;QUIT IF DONE
INX	H	;NOT DONE. STEP TO ANOTHER CHAR
MOV C,	M	;PICK IT UP
DCR	B	;COUNT THEM
PUSH	B	;SAVE COUNT AND POINTER OVER TYPEOUT
PUSH	H	
CALL	CTY01	;TYPE CHAR, WITH ~ PROCESSING
POP	H	
POP	E	
JMP	PSIN8	;TYPE WHOLE LINE
PSIN9:	POP B	;DONE RE-TYPING. RESTORE REAL COUNT
JMP	PSIN1	;AND GO GET ANOTHER CHAR

WERE ON ORDINARY CHAR.

```

PSIN10: INX      H          ;SAVE IN BUFFER
        MOV M, A
        INR     B          ;COUNT THEM
PSIN11: PUSH    B
        PUSH    H
        MOV C, A
        CALL    CTY01       ;TYPE CHAR
        PGP     H
        POP     B
        MOV A, M
        CPI    CH.ATN
        MOV A, B
        JNZ    PSIN12
        CPI    $01
        JZ     $0000
PSIN12: CMP    C          ;IF SO, WAS IT THE FIRST CHAR?
        JC     PSIN1       ;IF SO, RE-BOOT.
        ;UP TO FULL BUFFER?
        ;IF NOT, GET MORE.
PSIN13: POP    H          ;BACK TO START OF BUFFER
        MOV M, B          ;AND PUT THE COUNT THERE
        MVI C, CH.CR
        IHP    CTY01       ;ECHO THE C-R AT END
        ;TYPE CR

```

HPCS:	.BYTE 0	\$HORIZ POSITION OF TTY
LPTFLG:	.BYTE 0	\$NONZERO TO COPY TYO TO PRINTER
SAYCH:	.BYTE 0	\$BKJFN'ED TYI CHAR
CURDSX:	.BYTE 0	\$CURRENT DISK # (0-3)
USRFCN:	.BYTE 0	\$CALLER'S AC C, = BDOS FUNCTION
USERDE:	.BYTE 0,B	\$CALLER'S DE, = ARG ADDRESS
ANSWER:	.BYTE 0,C	\$ANSWER TO RETURN TO USER
USERSP:	.BYTE 0,B	\$USER'S STACK POINTER
	.BLKB 48	\$STACK AREA
BDOSSP:		\$BDOS HAS STACK ABOVE HERE

FRECHR: •BYTE \$E5

;FILL CHAR ON BLANK DISK, FOR DELETED FILES

PL

;HOME THE DISK AND SET THE TRACK NUMBER IN CORE TO ZERO

HOMSET: CALL CBHOME

;HOME THE DISK (BIOS). THIS IS THE
; ONLY SUCH CALL

LHLD TKBIAS

;GO TO LOGICAL TRACK ZERO

MOV C, L

;TO C FOR BIOS ARGUMENT

CALL CBSTTK

;SET TRACK NUMBER (BIOS) FROM C

LHLD TKSECB

;THAT'S SECTOR NUMBER ZERO, TOO.

MVI A, 0

; ..

MOV M, A

; ..

INX H

;ITS TWO BYTES EACH

MVI M, 0

;POINT TO TRACK NUMBER

LHLD TKPNTR

;CLEAR TRACK NUMBER

MOV M, 0

RET

COMPTK: LHLD TKSECB

;SECTOR 0 ON CURRENT TRACK

LXI D, CURRMW

;THIS CLUSTER

CALL DEMIHL

;(DE)-(HL)=>HL

JNC CMPTK1

;JUMP IF TEST SECTOR >= TRACK'S SECTOR

LHLD TKSECB

;BACK UP ONE TK. REDUCE (HL).

XCHG

;PUT IT IN DE

LDA SECPTK

;SECTORS PER TRACK

CALL DEMIDA

;(DE) - BA => HL

XCHG

DLX H

;CORRECT FOR MOD BY DEMIDA. BACK TO TKSECB
;PUT BACK REDUCED SECTOR NUMBER

MOV M, E

INX H

MOV M, D

;AND STORE BACK IN (DE)

LHLD TKPNTR

;POINT TO TRACK NUMBER

DCR M

;REDUCE TRACK NUMBER

JMP COMPTK

;LOOP TIL ON RIGHT TRACK

CMPTK1: LHLD TKSECB

;STEP UP ONE TRACK

LDA SECPTK

;SECTORS PER TRACK

CALL APLSMM

;(BA + (HL))=>HL

SHLD CTKTMR

;TEMP, NEW SECTOR

LXI D, CURRMW

;SECTOR NUMBER

CALL IDEMHL

;(DE) - HL => HL

JC CMPTK2

;JUMP IF DESIRED > TEST

LHLD TKSECB

;CURRENT REAL SECTOR 0 OF THIS TK

PUSH H

;TEST SECTOR

LHLD CTKTMR

XCHG

;STORE TEST AS CURRENT

PCP H

MOV M, E

INX H

MOV M, D

LHLD TKPNTR

;POINT TO TRACK NUMBER

INR M

;MAKE IT CORRESPOND TO SECTOR

JMP CMPTK1

;SEE IF THAT'S HIGH ENOUGH

CMPTK2: LHLD TKPNTR

;POINT TO TRACK NUMBER

LDA TKBIAS

;OFFSET ALL LOGICAL TRACKS FOR BOOT

ADD M

MOV C, A

;ARG TO BIOS

CALL CBSTTK

;SET TRACK (BIOS) FROM C

LHLD TKSECB

;SECTOR ZERO ON CHOSEN TRACK

```

LXI D, CURRMW      ;SECTOR NUMBER DESIRED
CALL DEMIHL        ;16 BIT SUBTRACT, TO HL
MOV C, L            ;BIOLOGICAL SECTOR ON THE TRACK
CALL SKEWSC        ;GET CORR'ING SECTOR FROM SKEW TBL,
RET                ;CALL BIOS WITH IT (ONLY SUCH CALL)

```

;COMMON ENTRY FOR READ AND WRITE. IF READ, C/1. IF WRITE, C/0.

```

RWCOM:  LXI H, RWARG      ;SAVE READ OR WRITE COMMAND HERE
       MOV M, C            ;READ OR WRITE COMMAND
       LDA RWARG           ;PICK IT BACK UP
       RAR                ;SEE WHICH IT IS
       JNC RWCMM          ;JUMP IF 0, FOR WRITE.
       CALL CBREAD         ;CALL BIOS TO READ A SECTOR.
; (THIS IS THE ONLY SUCH CALL)
       STA RWCOMT          ;STORE SUCCESS/FAIL CODE
       JMP RWCOMF          ;COMMON CLEANUP FOR RD AND WRITE.

```

```

RWCMM:  CALL CEWRIT      ;CALL BIOS TO WRITE A SECTOR.
; (THIS IS THE ONLY SUCH CALL)
       STA RWCOMT          ;STORE SUCCESS/FAIL CODE
RWCOMF:  LDA RWCOMT        ;SEE IF THERE WAS AN ERROR
       CPI 0               ;IF ERROR, GO HANDLE IT.
       JNZ RWCOME          ;IF NO ERROR, RETURN FROM RD/WRITE
       RET

```

```

RWCOME: LXI H, RWCOMZ      ;PLACE TO RETURN TO
       PUSH H              ;SAVE FOR LATER POPJ
       LHLD #EADSC          ;GET BAD SECTOR HANDLER (MAYBE PATCHED)
       PCHL                ;GO TO IT.
       RWCOMZ: RET          ;IF HANDLER COMES BACK, ACCEPT THE ERR

```

;ALL READS COME THROUGH HERE

```

RDSEC:  MVI C, $01          ;FLAG FOR READ
       CALL RWCOM          ;CALL COMMON I/O ROUTINE
       RET

```

;ALL WRITES COME THROUGH HERE

```

WRSEC:  MVI C, E            ;FLAG FOR WRITE
       CALL RWCOM          ;CALL COMMON I/O ROUTINE
       RET

```

;GET MAP BYTE FOR CURRENT NR

```

GTMAPB: LHLD LSECPC        ;CONSTANT, LOG SEC PER CLUSTER
       MOV C, L            ;33F8 21 D3 3D
       LXI H, CURRNR        ;GET M, DO (C) RAR'S, GIVING CLUSTER
       CALL RARNM          ;STEP TO MAP AREA OF FCB
       ADI $10              ;FOR THIS CLUSTER
       MOV C, A              ;AS ADDRESS (16 BITS)
       MVI B, 0              ;IN FCB
       LHLD USERDE          ;ADDR OF THIS MAP BYTE
       DAD B                ;THIS IS THE CLUSTER NUMBER
       MOV L, M              ;TREAT AS THOUGH COULD BE 16 BITS
       MVI H, 0              ;SAVE MAP BYTE (CLUSTER NUMBER)
       SHLD CURRMW          ;RET

```

;COMPUTE SECTOR NUMBER FROM CURRNR AND CURRMW

COMPSN:	LHLD	LSECPC	;SHIFT FACTOR, SECTORS/CLUSTER
	MOV C,	L	
	LXI H,	CURRMW	;GET THE CLUSTER NUMBER
	CALL	SHLMW	;MAKE A SECTOR NUMBER
	LDA	ALLMSK	;MASK FOR SECTOR IN CLUSTER
	PUSH	H	
	LXI H,	CURRNR	
	ANA	M	;MASK SECTOR WITHIN CLUSTER
	POP	H	;GET BACK CLUSTER AS SECTOR
	CALL	AORHL	;OR IN SECTOR WITHIN CLUSTER
	SHLD	CURRMW	;NOW HAVE FULL SECTOR NUMBER
	RET		

;GET NR AND RC FROM FCB

STNRRC:	LXI B,	\$0020	;GET NR BYTE
	LHLD	USERDE	;IN FCB
	DAD	B	
	MOV A,	M	;PICK IT UP
	STA	CURRNR	;SAVE RECORD NUMBER
	LXI B,	\$002F	;ALSO GET RC (REC COUNT)
	LHLD	USERDE	;FROM FCB
	DAD	B	
	MOV A,	M	;PICK IT UP
	STA	CURRRC	;SAVE RECORD COUNT
	RET		

;UPDATE NR AND RC IN FCB

UDNRRC:	LDA	CURRNR	;RECORD NUMBER
	INR	A	;STEP, PREPARE FOR NEXT RECORD
	LXI B,	\$0020	;POINT TO NR FIELD
	LHLD	USERDE	;IN USER'S FCB
	DAD	B	;STEP TO NR BYTE
	MOV M,	A	;STORE IT
	LXI B,	\$002F	;SAME FOR RC BYTE
	LHLD	USERDE	;IN FCB
	DAD	B	
	LDA	CURRRC	
	MOV M,	A	
	RET		;UPDATE IT.
			; 3451 69
			; 3452 3A D2 3D

GDIRSC:	LDA	DOSFCB	; 3457 3A F3 3C
	ANI	\$FE	;DIVIDE BY FOUR, GETS SECTOR NUMBER
	RAR		; FROM THE FCB NUMBER
	RAR		
STA	DIRSEC		;THIS IS THE DIRECTORY SECTOR NUMBER
	MOV C,	A	;MAKE A DOUBLE BYTE OF IT
	MVI B,	B	
	MOV H,	B	;COPY IT
	MOV L,	C	
	SHLD	CURRMW	;READY TO COMPUTE TK FOR THIS SECTOR
	CALL	COMPTK	;FIND TRACK AND SECTOR IN TRACK
	RET		

;CHECKSUM ROUTINE FOR DIRECTORY SECTORS. VERY INEFFICIENT. COMPILED?

GETCKS: LXI H, CKSTM2 ;SUM GOES HERE

Fri Mar 21 11:24:37 1980

/usr2/leo/cpm.asm

Page 1026

```
MVI M, 0
DCX H
MVI M, 0
GTCKS1: MVI A, NSECCH-1
LXI H, CKSTM1
CMP M
JC GTCKS2
LHLD CKSTM1
MVI H, 0
XCHG
LHLD SYSDMA           ;COMPUTE ADDRESS IN BUFFER
DAD D
LDA CKSTM2
ADD M
STA CKSTM2
LXI H, CKSTM1
INR M
JNZ GTCKS1
GTCKS2: LDA CKSTM2
RET
```

;CR ARG WITH BIT FOR CURR DISK

```
ORCURR: LXI H, ORCURT
MOV M, C
LDA CURDSK
INR A
MOV C, A
MVI A, $01
CALL RLCN
RAL
LXI H, ORCURT
CRA M
RET
```

;SAVE ARG
; ..
;NUMBER OF CURR DISK
;SINGLE BIT PATTERN
;DO (C) RLC'S
;BACK ONE
;GET CALLER'S ARG
;OR IT IN WITH NEW BIT

;GET R/O BIT FOR CURRENT DISK

```
CHKRO: LDA CURDSK
INR A
MOV C, A
LXI H, ROVEC
CALL RALNM
RLC
RET
```

;READ ONLY DISKS
;SHIFT IN RIGHT BIT
;PUT IT IN B

```
MAKERO: LHLD ROVEC
MOV C, L
CALL ORCURR
STA ROVEC
RET
```

;DISKS ALREADY LOCKED
;OR IN BIT FOR CURRENT DISK
;NOW IT'S LOCKED, TOO.

;CHECK R/O BIT AND GIVE ERR IF SET.

```
ERRRC: CALL CHKRO
RAR
JNC ERROX
LXI H, ERROX
PUSH H
```

;CHECK IT
;BIT INTO CY
;JUMP IF OK TO WRITE
;ELSE SET THIS RETURN PC
;ON STACK FOR RET

ERRROX: RET

L

VERDIR VERIFIES DIRECTORY SECTORS BY CHECKSUMS. CATCHES H/W ARE ERRORS,
AND CATCHES CHANGING DISKS OUT FROM UNDER BDOS.
C/0 SAYS CHECK IT AGAINST LAST VALUE. C/1 SAYS SET NEW VALUE.

VERDIR: LXI H, VERDIT	;LOCAL TEMP, CHECK CKSM OR NOT
MOV M, C	
LDA DIRSEC	;SECTOR FOR DIRECTORY
CPI \$10	;LEGAL SECTOR FOR DIR?
JC VERDI1	;JUMP IF SO.
RET	;IF NOT, PROB FF (FILE NOT FOUND)

VERDI1: LDA VERDIT	;SEE IF CKSUM KNOWN
RAR	
JNC VERDI2	;JUMP IF KNOWN
CALL GETCKS	;COMPUTE CHECKSUM OF THIS BUFFER
LHLD DIRSEC	;SECTOR IT CAME FROM
MVI H, 0	;16 BIT SECTOR NUMBER
XCHG	
LHLD VECCKS	;POINTER TO CHKSUM TABLE
DAD D	
MOV M, A	;HERE'S THE CHECKSUM, NOW KNOWN
JMP VERDI3	

VERDI2: LHLD DIRSEC	;KNOWN. GET SECTOR NUMBER
MVI H, B	;AS AN ADDR
XCHG	
LHLD VECCKS	;POINTER TO CHECKSUM TABLE
DAD D	;WHERE CKS OF THIS DIR SEC GOES
PUSH H	
CALL GETCKS	;COMPUTE CHECKSUM OF THIS BUFFER
POP H	;WHERE WE LAST PUT IT
CMP M	;IS IT THE SAME?
JZ VERDI3	;JUMP IF OK
CALL MAKERO	;MAKE DRIVE UNWRITABLE, DIRECTORY ERRORS
VERDI3: RET	

UPDATE DIRECTORY SECTOR ON DISK

DDDIR: MVI C, \$E1	;SET NEW CKSUM INTO TABLE
CALL VERDIR	; ..
CALL WRSEC	;WRITE A SECTOR
RET	

SET NEXT DIR SLOT NUMBER. GET ITS SECTOR IN CORE.

MAPNXT: LXI H, MAPNXV	;SAVE CHECK OR SET CKSUM FLAG
MOV M, C	
LDA DOSFCB	;LAST FCB NUMBER
INR A	;STEP TO NEXT ONE.
STA DOSFCB	;PUT IT BACK IN CORE
MVI C, A	;COPY IT
LDA MAXFCB	;HIGHEST LEGAL NUMBER?
CMP C	
JNC MAPNX1	;JUMP IF OK
LXI H, DOSFCB	;ELSE SAY NOT FOUND
MVI M, \$FF	
RET	

```

MAPNX1: LDA      DOSFCB           ;NEW FCB NUMBER
        ANI      $03              ;NUMBER WITHIN A SECTOR
        ADD      A                ;AS A BYTE NUMBER IN SECTOR
        ADD      A
        ADD      A
        ADD      A
        STA      CHROFS          ;CHARACTER OFFSET OF FCB IN DIR BUFFER
        CPI      0                ;FIRST FCB IN A SECTOR?
        JNZ      MAPNX2          ;IF NOT, HAVE IT IN CORE ALREADY
        CALL    GDIRSC          ;COMPUTE DIRECTORY SECTOR NEEDED
        CALL    RDSEC            ;READ A SECTOR
                                         ; 354C 2A DE 3D
        MOV C, L                ;CHECK OR SET CKSUM FLAG
        CALL    VERDIR           ;DO THE CHECKSUMMING

MAPNX2: RET

```

;GET ALLOC BIT FOR CLUSTER IN C

```

GETBTC: LXI H, GETBTT           ;CLUSTER NUMBER
        MOV M, C
        LDA      GETBTT          ;OVER 8 (BITS PER BYTE)
        ANI      $FC
        RAR
        RAR
        MCV C, A                ;AS AN ADDRESS
        MVI B, 0
        LHLD    CURALV          ;CURRENT DISK'S ALLOC VECTOR
        DAD    B                ;ADDR OF BIT IN TABLE
        LDA      GETBTT          ;NOW THE BIT NUMBER IN BYTE
        ANI      $07              ;BITS IN A BYTE
        INR    A                ;OFFSET BY ONE
        MOV C, A
        CALL    RLCNM            ;PICK UP (HL) INTO A, AND DO (C) RLC'S
        RET

```

;MARK THE BIT TABLE WITH THE BIT IN E FOR CLUSTER IN C

```

MRKBTB: LXI H, MRKBTW          ; 3572 21 E1 3D
        MOV M, E                ;SAVE CALLER'S C
        DCX    H
        MOV M, C                ;SAVE CLUSTER NUMBER
        LHLD    MRKBTV          ;CLUSTER NUMBER AGAIN
        MOV C, L                ;AS AN ADDRESS
        CALL    GETBTC          ;GET ALLOCATION BIT
        ANI      $FE              ;MASK OUT OLD VALUE
        LXI H, MRKBTW          ;POINT TO NEW VALUE
        ORA    M                ;DEPOSIT THE NEW BIT
        PUSH   PSW
        MVI A, $07              ;SAVE NEW BIT
        MVI A, $07              ;MASK FOR BIT NUMBER
        DCX    H                ;BACK TO THE CLUSTER NUMBER
        ANA    M                ;MASK FOR BIT IN BYTE
        INR    A                ;OFFSET BY ONE AGAIN
        MOV C, A                ;COPY FOR SUBR CONVENTION
        POP    PSW
        CALL    RALN            ;DO (C) RAL'S
        PUSH   PSW
        MOV A, M                ;SAVE IT, SHIFTED BACK
        MOV A, M                ;CLUSTER NUMBER AGAIN
        ANI      $0F              ;DIV BY 8 FOR A BYTE NUMBER

```

Fri Mar 21 11:25:24 1980

/usr2/leo/cpm.asm

Page 1.29

RAR

RAR
RAR

MOV C, A ;BYTE NUMBER MAKES ADDRESS AGAIN

MVI B, 0
LHLD CURALV ;CURRENT DISK'S ALLOC VECTOR
DAD B ;WHERE THE CURRENT BYTE LIVES
POP B
MOV C, B ;UPDATED BYTE
MOV M, C ;TO VECTOR
RET

SET OR CLEAR THIS FCB'S ALLOCATION IN BITTABLE

ALOFCB: LXI H, ALOFCV ;WILL SET BITS TO THIS (0 OR 1)
MOV M, C
LDA CHROFS ;CHARACTER OFFSET OF FCB IN DIR BUFFER
ADI \$1E ;MAP AREA
STA ALOFCW ;SAVE MAP ADDR
ALOFC1: LDA CHROFS ;CHARACTER OFFSET OF FCB IN DIR BUFFER
ADI \$1F ;SEE IF DONE
LXI H, ALOFCW ;WHERE WE ARE BY NOW
CMP M ;DONE?
JC ALOFC3 ;JUMP IF ALL DONE
LHLD ALOFCW ;NOT YET. POINT TO MAP BYTE
MVI H, E ;AS AN ADDRESS
XCHG
LHLD SYSDMA ;IN BUFFER
DAD D ;
MOV A, M ;GET A MAP BYTE
STA ALOFCX ;KEEP A COPY OF IT
CPI 0 ;IS THIS CLUSTER FREE?
JZ ALOFC2 ;IF SO, DON'T (DE)ALLOCATE IT
LHLD ALOFCX ;NO, MUST (DE)ALLOCATE IT
MOV C, L ;CLUSTER NUMBER
LHLD ALOFCV ;0 OR 1 FOR THAT CLUSTER
XCHG
CALL MRKSTB ;PUT BIT IN BIT TABLE
ALOFC2: LXI H, ALOFCW ;STEP TO NEXT BYTE OF MAP AREA
INR M
JNZ ALOFC1 ;LOOP UNTIL DONE

ALOFC3: RET

READ THE DIRECTORY AND ALLOCATE ALL FILES IN BIT TABLE

RDALOC: LXI H, ANSRL ;35E0 21 F0 30
MVI M, 0 ;35E3 36 00
LXI H, ROVEC ;35E5 21 7C 3D
MVI M, 0 ;SAY NO DISKS ARE R/O
LHLD CURALV ;CURRENT DISK'S ALLOC VECTOR
LDA INIALO
MOV M, A ;ALLOCATE THE DIRECTORY CLUSTERS
LXI H, RDALOV ;ADDR WITHIN ALLOC VECTOR
MVI M, \$01 ;START WITH BYTE AFTER ABOVE INIT
RDAL01: MVI A, NALLOV-1 ;DONE THEM ALL?
LXI H, RDALOV ;
CMP M ;
JC RDAL02 ;JUMP IF SO
LHLD RDALOV ;NO, DO ANOTHER

XCHG
 LHLD CURALV ;CURRENT DISK'S ALLOC VECTOR
 DAD D ;BYTE IN ALLOC VECTOR
 MVI M, 0 ;CLEAR ALLOCATION
 LXI H, RDALOV ;STEP THRU VECTOR
 INR M
 JNZ RDAL01 ;LOOP TILL DONE WHOLE VECTOR THIS DSK
 RDAL02: CALL HOMSET ;HOME THE DISK
 LXI H, DOSFCB ;SAY NO FCB NOW OPEN
 MVI M, \$FF
 RDAL03: MVI C, \$01 ;MAP AN FCB, SETTING UP ALLOC VEC
 CALL MAPNXT ;GET NEXT FCB NUM, MAP ITS SECTOR INTO CORE
 LDA DOSFCB ;DONE ALL FCB'S?
 CPI \$FF ;WHEN ALL DONE, RETURNS FF
 JNZ RDAL04 ;KEEP ON TILL HIGHEST LEGAL NUMBER
 RET

RDAL04: LHLD CHROFS ;CHARACTER OFFSET OF FCB IN DIR BUFFER
 MVI H, 2 ;COMPUTE ADDR IN BUFFER

LHLD SYS DMA
 DAD D
 MOV A, M ;GET FIRST CHAR OF FCB
 CPI \$E5 ;DISK FILL CHAR, = BLANK ENTRY
 JZ RDAL05 ;IF SO, THIS FCB IS A FREE ONE.
 LHLD CHROFS ;CHARACTER OFFSET OF FCB IN DIR BUFFER
 MVI H, 0 ;NOT FREE. COMPUTE UP THE NAME ADDR
 LXI B, \$2201 ;WHERE NAMES START

LHLD SYS DMA
 DAD D
 MOV A, M ;GET FIRST CHAR OF NAME
 SUI CH.DOL ;MAKE 8-BIT MATCH FLAG
 SUI \$01 ;A/FF IF CHAR WAS DOLLAR, ELSE A/
 LXI H, ANSARL ;SET ANSWER TO ONES IF ANY \$'S FOUND
 ORA M ;
 MOV M, A ;IN ANY CASE, ALLOCATE THE SPACE
 MVI C, \$01 ;SET ALLOCATION FOR THIS FCB IN BITTABLE
 CALL ALOFCB ;LOOP FOR ALL FCB'S

RDAL05: JMP RDAL03 .
 RET ;EXTRA RET FROM COMPILER

;GET NEXT MATCHING FILE

STPFIL: LHLD STPFIW ;USER'S FCB FOR PREV LOOKUP
 SHLD USERDE ;RE-USE IT
 STPFIS: MVI C, 0 ;REMEMBER TO CHECK CHECKSUMS
 CALL MAPNXT ;GET NEXT FCB NUM, MAP ITS SECTOR INTO CORE.
 LDA DOSFCB ;WHICH ONE DID IT FIND?
 STA ANSARL ;SAVE THE ANSWER
 CPI \$FF ;WERE ANY FOUND?
 JNZ STPFII ;JUMP IF SO
 RET ;RETURN IF NOT

STPFII: LXI H, STPFIX ;NUMBER CHARS CHECKED SO FAR
 MVI M, 0 ;INITIALLY, NONE CHECKED
 STPFIZ: LXI H, STPFIV ;# CHARS TO CHECK

	LDA	STPFIX	;NUMBER DONE
	SUB	M	;ALL DONE?
	SBB	A	;SET TO FF UNLESS ALL CHECKED, THEN 0
	LHLD	STPFIX	;CHAR POSITION IN FCB
	MVI H,	B	;AS AN ADDR
	XCHG		
LHLD	USERDE		
	DAD	D	
	PUSH	PSW	;SAVE ALL-CHECKED FLAG
	MOV A,	M	;GET THIS CHAR FROM USER FCB
	STA	STPFY	;SAVE USER'S CHAR
	LDA	STPFIX	;NUMBER CHECKED SO FAR
	PUSH	H	
	LXI H,	CHROFS	;CHARACTER OFFSET OF FCB IN DIR BUFFER.
	ADD	M	;CURRENT CHAR NUMBER IN BUFFER
	MOV C,	A	;AS ADDRESS
	MVI B,	B	
	LHLD	SYSDMA	;AS ADDR IN BUFFER
	DAD	B	
	POP	B	;BACK TO USER FCB
	LDAX	S	;CHAR FROM USER'S FCB
	SUB	M	;COMPARE THEM
	SUI	\$D1	;IF MATCH, SET CY
	SBB	A	;AND MAKE FF IF MATCH, ELSE 0
	PUSH	PSW	;SAVE MATCHED CHAR FLAG
	LDA	STPFY	;USER'S CHAR
	SUI	\$3F	;QUESTIONMARK? (WILDCARD)
	SUI	\$D1	;MAKE FLAG FOR THAT
	SBB	A	
	POP	B	;MATCHED CHAR THIS TIME FLAG
	MOV C,	B	
	CRA	C	;ONES IF MATCHED OR WILDCARD
	POP	B	
	MOV C,	B	;MORE CHARS LEFT FLAG
	ANA	C	
	RAR		
JNC	STPF13		;JUMP IF ALL DONE OR DIDNT MATCH
	LXI H,	STPFIX	;HAVE TO CHECK SOME MORE
	INR	M	;STEP TO NEXT CHAR NUMBER
	JMP	STPF12	;GO DO ANOTHER CHAR

;HERE IF NON-MATCH OR ALL DONE.

STPF13:	LXI H,	STPFIV	;NUMBER CHARS TO CHECK
	LDA	STPFIX	;NUMBER CHECKED
	CMP	M	;ALL DONE?
	JNZ	STPF14	;IF NOT, NON-MATCH. QUIT.
	RET		;ELSE ALL OK, SO RETURN
STPF14:	JMP	STPF15	;GO TRY ANOTHER FCB
	RET		;EXTRA RET FROM COMPILER

;ROUTINE TO LOOK FOR A FILENAME, OR SPECIFIC EXTENT.

LOOKUP:	LXI H,	LOOKUV	;SAVE NUMBER CHAR TO VERIFY
	MOV M,	C	;HERE
	LDA	LOOKUV	
	STA	STPFIV	;SAVE FOR LATER SCAN
	LHLD	USERDE	;COPY USER'S FCB ADDR

```

SHLD    STPF1W      ;FOR LATER
LXI H, DOSFCB   ;ASSUME FAILURE
MVI M, $FF       ;FLAGGED BY FF
CALL    HOMSET    ;HOME AND SET TRACK
CALL    STPF1L    ;GET NEXT MATCHING FILE
RET

```

;FUNCTION ROUTINE FOR DELETE FILE CALLS THIS TO DO THE WORK.

```

DODEL:  CALL    ERRO      ;CHECK R/O AND ERROR OUT IF SET
        MVI C, $0C      ;CHECK THIS MANY CHARS
        CALL    LOOKUP    ;FIND USER'S FILE
DODEL1: LDA    DOSFCB    ;WAS THERE ONE?
        CPI    $FF
        JNZ    DODEL2    ;JUMP IF SO
        RET             ;ELSE JUST RETURN

```

;DOING A DELETED AND THERE IS A FILE TO DO.

```

DODEL2: MVI C, 0
        CALL    ALGFCB    ;DEALLOCATE FLAG FOR BITTABLE
        LHLD    CHROFS    ;CLEAR ALLOC FOR THIS FCB IN BITTABLE
        MVI H, 0           ;CHARACTER OFFSET OF FCB IN DIR BUFFER
        YCHG
LHLD    SYSDMA
        DAD    0
        MVI M, $E5      ;AS AN ADDR
        CALL    UPDIR     ;DATA FILL CHAR, MAKING FREE FCB
        CALL    STPF1L    ;UPDATE DIR SEC ON DISK
        JMP    DODEL1    ;GET NEXT MATCHING FILE OR EXTENT
        RET             ;LOOP FOR ALL
                           ;EXTRA RET FROM COMPILER

```

TL

;FROM WRITE NEXT RECORD ROUTINE
; FIND A FREE CLUSTER NEAR CLUSTER IN C.

```

ALLOC:  LXI H, ALO.LO    ;SAVE DESIRED CLUSTER
        MOV M, C
        LDA    ALO.LO    ;AS BOTH LOW AND HIGH TESTS
        STA    ALO.HI
ALLOC1: LXI H, MXCLUS    ;NUMBER OF CLUSTERS ON DISK
        LDA    ALO.HI    ;TEST HIGH FOR BEING OFF TOP
        SUB    M
        SBB    A
        PUSH   PSW
        MVI A, 0
        LXI H, ALO.LO
        SUB    M
        SBB    A
        PCP    B
        MOV C, B
        ORA    C
        RAR
JNC    ALLOC4    ;JUMP IF NEITHER ONE GOOD
        MVI A, 0
        LXI H, ALO.LO
        SUB    M
        SBB    A
        ANI    $01
        PUSH   PSW
                           ;NOW SEE WHICH IS OK.
                           ;BY WASTEFULLY RECOMPUTING
                           ;WHETHER .LO IS LEGAL
                           ;1 IF NOT ZERO

```

```

MOV A, M           ;LO
POP B
MOV C, B           ;1 IF NOT FREE
SUB C             ;DESIRED -(1 OR 0)
MOV M, A           ;STORE NEW TARGET
INX H             ;ALO.HI
MOV A, M
LXI H, MXCLUS    ;NUMBER OF CLUSTERS ON DISK
SUB M             ;IS HIGH AT MAX?
SBB A             ;IF LEGAL, MAKE FF
ANI $01            ;1 IF LEGAL, & IF NOT
LXI H, ALO.HI     ;STEP HI UP IF LEGAL
ADD M             ; ..
MOV M, A           ; ..
LHLD ALO.HI       ;SEE IF .LO IS FREE
MOV C, L
CALL GETBTC       ;GET ITS ALLOCATION BIT
RAR
JC    ALLOC2       ;TEST IT
JUMP IF NOT FREE
LDA   ALO.HI       ;IT'S FREE, USE IT.
RET

```

L

; MORE OF SEARCH FOR FREE CLUSTER

```

ALLOC2: LHLD ALO.LO      ;SEE IF THE LOW SIDE GUY IS FREE
MOV C, L
CALL GETBTC        ;GET ITS ALLOC BIT
RAR
JC    ALLOC3       ;JUMP IF NOT FREE
LDA   ALO.LO       ;FREE. USE IT.
RET

```

```

ALLOC3: JMP ALLOC1      ;GO BACK AND TRY AGAIN

```

```

ALLOC4: MVI A, B        ;THERE ARE NO FREE CLUSTERS
RET

```

;COPY NAME STRING INTO SYSTEM DIRECTORY. FOR MAKE OR RENAME.

```

CPYNAM: LXI H, CPYNAM    ;TWO ARGUMENTS
MOV M, E             ;COUNT OF CHARS TO MOVE
DCX H
MOV M, C             ;WHERE IN USER FCB TO GET CHARS
CPYNM1: LDA CPYNAM     ;GET THE COUNT
DCR A               ;COUNT IT DOWN
STA CPYNAM          ;IN CORE, TOO
CPI $FF              ;ALL DONE?
JZ    CPYNM2         ;JUMP IF SO
LDA CPYNAM          ;IF NOT, GO GET ANOTHER CHAR FROM FCB
LXI H, CPYNAM        ;AT THIS OFFSET IN USER FCB
ADD M
MOV C, A             ;MAKE 16 BIT ADDR
MVI B, 0
LHLD USERDDE        ;IN FCB
DAD B
LDA CHROFS          ;CHARACTER OFFSET OF FCB IN DIR BUFFER
PUSH H               ;SAVE WHERE USER'S CHAR IS
LXI H, CPYNAM        ;COMPUTE WHERE IT IS IN DIR BUFFER
ADD M
MOV C, A             ;AGAIN, 16 BIT ARITHMETIC

```

```

MVI B, 0
LHLD SYSDMA
DAD B
POP B
LDAX B
MOV M, A
JMP CPYNM1

;IN SYSTEM BUFFER
;WHERE USER CHAR IS
;GET USER CHAR
;PUT IN SYSTEM DIR
;SEE IF ALL DONE

CPYNM2: CALL GDIRSC
CALL UPDDIR
RET

```

TL
;COPY A WHOLE FCB

```

CPYFCB: MVI E, $20
MVI C, 0
CALL CPYNAM
RET

;COPY THIS MANY CHAR'S
;AT THIS OFFSET
;COPY USER NAME INTO DIR

```

;RENAME FILE ROUTINE

```

DOREN: CALL ERR0
MVI C, $0C
CALL LOOKUP
LHLD USERDE
LXI B, $2014
PUSH H
LHLD USERDE
DAD B
POP D
LDAX D
MOV M, A
;CHECK R/O AND ERROR OUT IF SET
;CHECK THIS MANY CHARS
;TRY TO FIND A FILE
;POINT TO NEW NAME
;IT STARTS HERE
;SAVE START OF FCB

DOREN1: LDA DOSFCB
CPI $FF
JZ DOREN2
MVI E, $0C
MVI C, $10
CALL CPYNAM
CALL STPFIL
JMP DOREN1
;FIND NEW NAME
;COPY ENTRY TYPE BYTE

;GET IT
;THERE WAS A FILE, WASN'T THERE?
;**
;JUMP IF NOT
;COPY THIS MANY CHARS
;FROM HERE IN USER'S FCB
;COPY NEW NAME INTO DISK DIR
;GET NEXT MATCHING FILE OR EXTENT
;LOOP TILL ALL FCB'S FOUND

```

```

DOREN2: RET

```

TL
;OPEN FILE ROUTINE. DO LOOKUP, COPY MAP AREA FROM DISK DIR TO FCB.

```

OPENFI: MVI C, $0D
CALL LOOKUP
LDA DOSFCB
CPI $FF
JZ OPENNF
LXI H, OPNFI4
MVI M, $0D
;CHECK THIS MANY CHARS
;GO FIND USER'S NAME
;GET ANSWER
;WAS IT THERE?
;IF NOT, GO GIVE ERROR
;COPY MAP AREA INTO FCB IN USER SPACE
;INITIAL INDEX

OPNFI1: MVI A, $1F
LXI H, OPNFI4
CMP M
JC OPENNF
LDA OPNFI4
LXI H, CHROFS
ADD M
;END TEST. DONE THEM ALL?
;**
;**
;JUMP IF DONE
;MORE TO MOVE. BUILD ADDRESS
;CHARACTER OFFSET OF FCB IN DIR BUFFER

```

```

MVI B, 0
LHLD SYSDMA      ;IN SYSTEM'S DIR COPY
DAD B
PUSH H
LHLD OPNFIV      ;MOVE IT TO SAME PLACE IN USER FCB
MVI H, 0
XCHG
LHLD USERDE      ;WHICH IS HERE
DAD D
POP B
LDAX B
MOV M, A
LXI H, OPNFIV
INR M
JNZ OPNFIV
OPENNF: RET

;CLOSE FILE LOGIC

```

```

CLOSFI: LXI H, ANSRL
        MVI M, 0
        CALL CHKRO
        RAR
        JNC CLSF11
        RET

        ;WHERE USER GETS ANSWER
        ;SUCCESS FLAG
        ;IS DISK WRITABLE?
        ; ..
        ;JUMP IF SO
        ;RETURN IF NOT. IT'S CLOSED.

CLSF11: MVI C, $20
        CALL LOOKUP
        LDA DOSFCB
        CPI $FF
        JZ CLSF12
        CALL CPYFCB
CLSF12: RET

        ;THIS MANY CHARS
        ;GET THE FCB
        ;THERE WAS ONE, YES?

        ;JUMP IF NO
        ;COPY WHOLE FCB, UPDATING SIZE, ALLOC...

```

;MAKE FILE ROUTINE

```

MAKIFI: CALL ERRO
        LHLD USERDE
        SHLD MAKFIW
        LXI H, FRECHR
        SHLD USERDE
        MVI C, $01
        CALL LOOKUP
        LDA DOSFCB
        CPI $FF
        JZ MAKFI3
        LHLD MAKFIW
        SHLD USERDE
        LXI H, MAKFIV
        MVI M, $0D
MAKFI1: MVI A, $1F
        LXI H, MAKFIV
        CMP M
        JC MAKFI2
        LHLD MAKFIV
        MVI H, 0
        XCHG
LHLD USERDE      ;USER'S FCB ADDR
DAD D
MVI M, 0          ;CLEAR THESE CHARS

```

```

LXI H, MAKFI4      ;DO THEM ALL
INR M              ;STEP THRU THEM
JNZ MAKFI1         ;LOOP TILL ALL DONE
MAKFI2: CALL CPYFCB ;COPY WHOLE FCB TO DISK BUFFER
MAKFI3: RET

```

;STEP TO THE NEXT EXTENT

```

STPXTN: LXI H, STPXTV    ;SAVE RD/WR FLAG
      MOV M, C
      CALL CLOSF1
      LDA DOSFCB
      CPI $FF
      JNZ STPXT1
      RET

```

;CLOSE THE OLD EXTENT
;DID THAT WORK?
; ..
;JUMP IF SO
;FAILED TO CLOSE.

```

STPXT1: LXI B, $B00C
      LHLD USERDE
      DAD B
      INR M
      MVI C, $00
      CALL LOOKUP
      LDA DOSFCB
      CPI $FF
      JNZ STPXT3
      LDA STPXTV
      RAR
      JNC STPXT2
      RET

```

;EXTENT BYTE
;IN USER'S FCB
;FOR FILE JUST CLOSED
;STEP TO NEXT EXTENT
;LOOK FOR THAT EXTENT
; ..
;FIND IT?

;JUMP IF SO
;NOT FOUND. WHAT NEXT?
;WERE WE READING OR WRITING?
;JUMP IF WRITING
;READING. JUST EOF, THEN.

```

STPXT2: CALL MAKEFI
      JMP STPXT4

```

;WRITING. MAKE ANOTHER EXTENT.
;IT'S OPEN.

```

STPXT3: CALL OPENFI
STPXT4: LDA DOSFCB
      CPI $FF
      JNZ STPXT5
      LXI H, ANSARL
      MVI M, $01
      RET

```

;OPEN THE NEXT EXTENT
;OPENED OK?

;JUMP IF SO
;NOT ABLE TO OPEN NEXT EXTENT.
;GIVE THIS ERROR RETVALUE
;RETURN TO USER

```

STPXT5: CALL STNRRC
      LXI H, ANSARL
      MVI M, 0
      RET

```

;GET NR AND RC FROM FCB
;USER ANSWER
;IS SUCCESS

```

SETDMA: LDA FRZDMA
      RAR
      JNC SETDM1
      LHLD USRDMA
      MOV B, H
      MOV C, L
      CALL CBSTMA
      RET

```

;DMA FROZEN?
;JUMP IF SO
;ALLOWED TO MOVE.
;SET TO WHAT USER LAST ASKED FOR
;BIOS SET DMA ADDRESS

;RESET DMA TO STANDARD

```

RSTDMA: LDA FRZDMA
      RAR
      INC RSTDMDY

```

;ALLOWED TO MOVE DMA?
;IF FROZEN, SKIP THIS.

LHLD	SYSDMA	;OK TO MOVE IT.	
MOV B,	H	3	38F0 44
MOV C,	L	3	38F1 40
CALL	CBSTMA	;BIOS SET DMA ADDRESS	
RSTDMA:	RET	3	38F5 C9

;HERE TO READ NEXT RECORD

RDNXTR:	CALL	STNRRC	;GET NR AND RC FROM FCB	
	LDA	CURRNR	;WHERE WERE WE?	
	LXI H,	CURRRC	;HOW FAR CAN WE GO?	
	CMP	M	;OVERDONE IT?	
	JC	RDNXT2	;JUMP IF OK	
	LXI H,	ANSARL	;IF TOO FAR, GIVE EOF RETURN	
	MVI M,	\$01	;TO USER'S ANSWER	
	LDA	CURRNR	;WERE WE TO END OF EXTENT?	
	CPI	\$8B	; ..	
	JNZ	RDNXT1	;JUMP IF NOT	
	MVI C,	\$01	;YES. SAY READING	
	CALL	STPXIN	;AND TRY FOR ANOTHER EXTENT	
RDNXT1:	LXI H,	CURRNR	;SAY FIRST RECORD IN EXTENT	
	MVI M,	0	;WHAT WAS ANSWER?	
	LDA	ANSARL	;SUCCESS?	
	CPI	0	;IF SO, MORE TO DO.	
	JZ	RDNXT2	;ELSE GIVE UP HERE.	
	RET			
RDNXT2:	CALL	GTMAPB	;GET MAP BYTE FOR CURRENT NR	
	LHLD	CURRMW	;THAT'S CLUSTER NUMBER	
	MOV A,	L	;AS 8 BIT NUMBER	
	CPI	E	;IS IT ALLOCATED?	
	JNZ	RDNXT3	;JUMP IF SO	
	LXI H,	ANSARL	;NOT ALLOCATED. SAY EOF.	
	MVI M,	\$01	;ALLOCATION CAN'T BE SPARSE.	
	JMP	RDNXT4	;SO QUIT ON FIRST ZERO	
RDNXT3:	CALL	COMPSN	;GET SECTOR NUMBER, FROM CURRMW, CURRNR	
	CALL	COMPTK	;AND COMPUTE THE TRACK FROM THAT	
	CALL	SETDMA	;SET UP TO USER'S (OR STD) DMA ADDR	
	CALL	RDSEC	;READ A SECTOR	
	CALL	RSTDMA	;BACK TO WHAT USER SAID	
	CALL	UDNRRC	;UPDATE NR AND RC IN FCB	
RDNXT4:	RET		;END OF READ-NEXT-RECORD	

;WRITE NEXT RECORD ROUTINE

WTNXTR:	CALL	ERRD	;CHECK R/D AND ERROR OUT IF SET	
	CALL	STNRRC	;GET NR AND RC FROM FCB	
	MVI A,	\$7F	;ABOUT TO GO OFF END OF EXTENT?	
	LXI H,	CURRNR	;WEREN'T ABLE TO MAKE A NEW EXTENT, THEN.	
	CMP	M	; ..	
	JNC	WTNXT1	;JUMP IF WITHIN AN EXT	
	LXI H,	ANSARL	;REPLY FULL.	
	MVI M,	\$01	;COULDN'T EXTEND FILE.	
	JMP	WTNXT8	;RETURN TO USER	
WTNXT1:	CALL	GTMAPB	;GET CLUSTER NUMBER FROM MAP	
	LHLD	CURRMW	;PICK IT UP	
	MOV A,	L		

```

CPI    0           ;IS IT ALLOCATED?
JNZ    WTNXT4      ;JUMP IF SO
LXI H, WTNXTV      ;NO, MUST ALLOCATE IT
MVI M, 0           ;PREV ALLOC = 0
LDA    CURRNR      ;CONVERT TO CLUSTER NUMBER
ANI    $FC          ;BY DIVIDING BY 8
RAR

RAR
ADI   $10          ;STEP INTO FCB TO MAP AREA
INX    H           ;TO WTNXTW
MOV    M, A         ;SAVE FCB OFFSET FOR THIS CLUSTER
MOV    C, A         ;IS IT THE INITIAL ONE OF FILE?
MVI    A, $10        ;  ..
CMP    C           ;JUMP IF SO
JNC    WTNXT2      ;AFTER FIRST CLUSTER
LDA    WTNXTW      ;BACK UP ONE, SEE WHERE IT WAS
DCR    A           ;GET THAT MAP BYTE
MOV    C, A         ;
MVI    B, B         ;
LHLD   USERDE      ;OFFSET INTO USER'S FCB
DAD    B           ;FINALLY, GET CLUSTER NUMBER
MOV    A, M         ;STORE IT
STA    WTNXTV      ;NOW GET DESIRED NEIGHBOR
WTNXT2: LHLD   WTNXTV      ;IN RIGHT AC FOR SUBR
MOV    C, L         ;ALLOCATE A CLUSTER NEAR IT
CALL   ALLOC        ;HERE'S THE ONE WE CAN HAVE
STA    WTNXTV      ;OR WAS IT FULL?
CPI    E           ;JUMP IF OK
JNZ    WTNXT3      ;FULL. USER'S ANSWER CELL
LXI H, ANSARL      ;PUT FULL FLAG IN IT
MVI M, $02          ;GO FINISH UP
JMP    WTNXT4

WTNXT3: LHLD   WTNXTV      ;HERE IS THE NEW CLUSTER
MOV    C, L         ;PUT A ONE IN BIT TABLE
MVI E, $01          ;  ..
CALL   MRKBTB      ;SAVE AS CURRENT CLUSTER
LHLD   WTNXTV      ;FULL WORD QUANTITY
MVI H, B           ;  ..
SHLD   CURRMW      ;HERE'S THE FCB SLOT WHICH GETS IT
LHLD   WTNXTW      ;FIGURE ADDRESS FOR IT
MVI H, B           ;
XCHG

LHLD   USERDE      ;IN USER'S FCB
DAD    D           ;THE CLUSTER NUMBER AGAIN
LDA    WTNXTV      ;INTO THE FCB
MOV    M, A         ;WAS THERE AN ERROR YET?
WTNXT4: LDA    ANSARL      ;  ..
CPI    B           ;YES, SO DON'T WRITE.
JNZ    WTNXT8      ;GET SECTOR NUMBER, FROM CURRMW, CURRNR
CALL   COMPSN      ;AND THE TRACK FROM THAT
CALL   COMPTK      ;SET UP TO USER'S (OR STD) DMA ADDR
CALL   SETDMA      ;WRITE A SECTOR
CALL   WRSEC       ;BACK TO WHAT USER SAID
LDA    CURRN R     ;THIS RECORD
LXI H, CURRRC      ;UP TO COUNT IN FCB?
CMP    M           ;  ..

```

Fri Mar 21 11:28:01 1980

/usr2/leo/cpm.asm

Page 1-39

```
LDA    CURRNR      ;YES. CURRENT PLUS 1 TO MAX
INR    A
STA    CURRRC      ;FOR LATER WRITING ON DISK
JNXT5: LDA    CURRNR      ;DID WE JUST FILL EXTENT?
CPI    $7F
JNZ    WTNXT7      ;NORMALLY, NO.
CALL   UDNRRRC    ;UPDATE NR AND RC IN FCB
MVI   C, 0
CALL   STPXTN     ;SAY WRITING
LDA    ANSARL      ;AND TRY FOR ANOTHER EXTENT
CPI    0
JNZ    WTNXT6      ;WAS IT SUCCESSFUL?
LXI   H, CURRNR    ;JUMP IF SO
MVI   M, $FF        ;PUT A BAD NR, FOR NEXT WRITE.
NXT6: LXI   H, ANSARL    ;WILL GET EOF IF TRIES AGAIN.
MVI   B, 0
NXT7: CALL  UDNRRRC    ;SAY THIS WRITE WAS OK, THOUGH.
NXT8: RET          ;UPDATE NR AND RC IN FCB
```

LOG IN THE DISK IN CURDSK

```
LOGCUR: MVI   A, $23      ;RANGE CHECK THE DRIVE NUMBER
LXI   H, CURDSK    ;CHECK USER'S ARG
CMP   M
JNC   LCGCU1      ;LEGAL?
LXI   H, LOGCU1    ;JUMP IF OK
PUSH  H
LHLD  VSELER      ;HANDLE BAD UNIT
PCHL

LCGCU1: LDA    CURDSK    ;DISK NUMBER
ADD   A
ADD   A
ADD   A
ADD   A
ADD   A
MOV   C, A
MVI   B, 0
LXI   H, ALLOCS    ;DISK NUMBER * 16
DAD   B
SHLD  CURALV      ;TIMES 32 BYTES PER DISK
LDA    CURDSK
ADD   A
ADD   A
ADD   A
ADD   A
MOV   C, A
MVI   B, 0
LXI   H, CKSTAB    ;MAKE ADDRESS
DAD   B
SHLD  VECCKS      ;BASE OF ALLOC AREA
LHLD  CURDSK      ;FOR THIS DISK
MVI   H, 0
LXI   B, TKPTRS    ;CURRENT DISK'S ALLOC VECTOR
DAD   B
SHLD  TKPNTR      ;DISK NUMBER * 16
LHLD  CURDSK
MVI   H, 0
LXI   B, TSEC0V    ;AS AN ADDRESS
DAD   B
SHLD  TKPNTR      ;SPACE FOR CHECKSUMS
LHLD  CURDSK      ;BASE FOR CKSM'S OF THIS DISK'S DIR
MVI   H, 0
LXI   B, TKPTRS    ;POINTER TO CKSUM. TABLE
DAD   B
SHLD  CURDSK      ;COMPUTE POSITION IN TRK POINTERS TABLE
MVI   H, 0
LXI   B, TKPTRS    ;FOR THIS DISK
DAD   B
SHLD  TKPNTR      ;..
LHLD  CURDSK
MVI   H, 0
LXI   B, TSEC0V    ;DEFINE WHERE TRACK NUMBER LIVES
DAD   B
SHLD  TKPNTR      ;FOR CURRENT DISK
MVI   H, 0
LXI   B, TSEC0V    ;SIMILARLY FOR SECTOR 0 ON THAT TK
DAD   B
SHLD  TKPNTR      ;VECTOR OF SECTOR 0 ON TK
MVI   H, 0
LXI   B, TSEC0V    ;DOUBLE THE DISK NUMBER, TO MAKE WORD
```

DAD	B	;IN THIS TABLE
SHLD	TKSEC0	;NOW LET I/O PACKAGE KNOW IT
LHLD	CURDSK	;BIOS - SELECT A DISK. THIS IS
HV C,	L	;CURRENTLY LOGGED DISKS
CALL	CBSELE	;SEE IF IT WAS KNOWN ALREADY
; THE ONLY SUCH CALL		
LDA	LOGVEC	
RLC		
PUSH	PSW	
LDA	CURDSK	
INR	A	
MOV C,	A	
POP	PSW	
CALL	RALN	
RAR		
JC	LOGCU2	
LHLD	LOGVEC	
MOV C,	L	
CALL	ORCURR	
STA	LOGVEC	
CALL	RDALLOC	
LOGCU2:	RET	

;LOG IN AND SELECT A DISK

LGNDISK:	LXI H, CURDSK	
LDA	ANSARG	;NEWLY REQUESTED DISK
CMP	M	;SAME DISK AS BEFORE?
JZ	LGNDISK1	;YES.
LDA	ANSARG	;NO, SO CHANGE TO IT
STA	CURDSK	
CALL	LOGCUR	
LGNDISK1:	RET	

SETDSK:	LHLD USERDE	;GET FCB ADDR
MVI A,	\$1F	;LOW 5 BITS FOR DRIVE
ANA	X	;ENTRY TYPE FIELD. USUALLY 0.
DCR	A	;CONVERT TO 0-3
STA	ANSARG	;DISK USER HAS IN FCB
CPI	\$1E	;DID USER SPECIFY ONE?
JNC	SETDSK1	;NORMALLY ZERO, THAT'S ALL.
LDA	CURDSK	
STA	RELOGN	
LHLD	USERDE	
MOV A,	M	
STA	RELOGF	
LHLD	USERDE	
MVI A,	\$E0	
ANA	M	
HV M,	A	
CALL	LGNDISK	
SETDSK1:	RET	

;SET DMA FROM HERE. ONE CALL, SETS TO DEFBFR

DEFDMA:	LHLD USERDE	;GET SYSTEM'S DMA ADDR ARG
SHLD	SYSDMA	;SAVE IN THIS SEMI-CONSTANT
MOV B,	H	;SWITCH REGISTERS FOR BIOS
MOV C,	L	
CALL	CBSTMA	;VECTOR SET DMA (BIOS)

RET

3WORK ROUTINE FOR USER'S CALLS, FROM BDOSH, FROM 5

BDOS00:	LHLD	USERDE	
	MOV A,	L	;USER'S "E"
	STA	ANSARG	;SAVE IT
	LXI H,	E	
	SHLD	ANSWER	;DEFAULT ANSWER IS "00,00"
	MOV A,	L	
	STA	ANSARL	;J TO "A" ANSWER (SEE T353)
	STA	RELOGF	;ASSUME NO NEED TO RE-LOG
	LHLD	USRFCN	;GET USER'S FUNCTION CODE
	MOV C,	L	;..
	MVI B,	B	;AS A FULL ADDRESS
	LXI H,	FCNDTB	;COMPUTE DISP TABLE ADDRESS
	DAD	B	;..
	DAD	B	;..
	MOV E,	M	;PICK UP THE ROUTINE ADDR
	INX	H	
	MOV D,	M	;TWO BYTES
	XCHG		;AND GO TO IT.
	PCHL		;..

3HERE ARE THE FUNCTION HANDLERS, FCN0 THRU FCN30

FCN0:	CALL	CBMBUT	;DO A WARM RE-BOOT, IN BIOS
	JMP	MRETN	
FCN1:	CALL	RDCONS	;READ CONSOLE CHAR
	STA	ANSARL	;SAVE ANSWER
	JMP	MRETN	
FCN2:	LHLD	ANSARG	;TYPE ROUTINE
	MOV C,	L	;USER'S D
	CALL	CTY02	;TYPE, WITH TAB SIMULATION
	JMP	MRETN	
FCN3:	CALL	CBPTRI	;READ FROM "READER", BIOS
	STA	ANSARL	;SAVE ANSWER
	JMP	MRETN	
FCN4:	LHLD	ANSARG	;WRITE ON "PUNCH"
	MOV C,	L	;CHAR TO WRITE
	CALL	CBPTPO	;IN BIOS
	JMP	MRETN	
FCN5:	LHLD	ANSARG	;WRITE ON "LISTING"
	MOV C,	L	;CHAR TO LIST
	CALL	CBLPTO	;IN BIOS
	JMP	MRETN	
FCN6:	LHLD	•BDOS•+1	;GET ADDRESS OF BDOS
	SHLD	ANSWER	;FOR USER
	JMP	MRETN	
FCN7:	LHLD	\$0003	;GET I/O STATUS BYTE
	MVI H,	E	;ONLY THE "L" PART
	SHLD	ANSWER	;SAVE AS ANSWER

	JMP	MRETN	
FON8:	LDA	ANSARG	;SET I/O STATUS BYTE
	STA	\$0003	;EASY?
	JMP	MRETN	
FON9:	LHLD	USERDE	;PRINT STRING, TERM BY "\$"
	MOV B,	H	;SWAP AC'S
	MOV C,	L	
	CALL	DLSOUT	;WE HAVE A SUBROUTINE FOR THAT
	JMP	MRETN	
FON10:	CALL	PSIN	;READ STRING, BUFFERED (PSIN)
	JMP	MRETN	
FON11:	CALL	CHKKBD	;CHECK KEYBOARD FLAG
	STA	ANSARL	;SAVE ANSWER
	JMP	MRETN	
FON12:	JMP	MRETN	;LIFT HEAD (NOP)
FON13:	LXI H,	CURDSK	;INITIALIZE DDOS
	MVI M,	0	; 3B6A 36 03
	LXI H,	LOGVEC	; 3B6C 21 C1 3D
	MVI M,	2	; 3B6F 30 00
	LXI H,	FRZDMA	;SAY ALLOWED TO MOVE DMA AROUND
	MVI M,	0	; ..
	LXI H,	\$0080	;DEFAULT DMA ADDR
	SHLD	USERDE	; 3B79 22 0A 33
	CALL	DEFDMA	; 3B7C CD C7 3A
	CALL	LOGCUR	; 3B7F CD 18 3A
	JMP	MRETN	
FON14:	CALL	LGNDISK	;SELECT AND LOG IN DISK
	JMP	MRETN	
FON15:	CALL	SETDSK	;OPEN FILE
	CALL	OPENFI	
	JMP	MRETN	
FON16:	CALL	SETDSK	;CLOSE FILE
	CALL	CLOSFI	; 3B97 CD 24 38
	JMP	MRETN	
FON17:	CALL	SETDSK	;SEARCH FOR FILENAME
	MVI C,	\$0D	;CHECK 13 CHARS
	CALL	LOOKUP	
	JMP	MRETN	
FON18:	LHLD	STPFIW	;SEARCH FOR NEXT OCCURENCE OF NAME
	SHLD	USERDE	;USERS FCB FROM PREV LOOKUP
	CALL	SETDSK	;PROCESS ENTRY TYPE FOR DRIVE
	CALL	STPFIL	;GET NEXT MATCHING FILE
	JMP	MRETN	;ANSWER IS IN ANSARL
FON19:	CALL	SETDSK	;DELETE FILE
	CALL	DODEL	; 3BBB CD E7 36
	JMP	MRETN	
FON20:	CALL	SETDSK	;READ NEXT RECORD

	CALL RDNXTR JMP MRETN		3 3BC3 CD F6 38
FCN21:	CALL SETDSK CALL WTNXTR JMP MRETN	; WRITE NEXT RECORD	3 3BCC CD 4A 39
FCN22:	CALL SETDSK CALL MAKEFI JMP MRETN	; MAKE FILE	3 3BD5 CD 42 38
FCN23:	CALL SETDSK CALL DOREN JMP MRETN	; RENAME FILE	3 3BDE CD B9 37
FCN24:	LDA LOGVEC STA ANSARL JMP MRETN	; INTERROGATE LOGIN VECTOR ; WHICH IS JUST THESE BITS	
FCN25:	LDA CURDSK STA ANSARL JMP MRETN	; GET CURRENT DRIVE NUMBER ; SAVE AS ANSWER	
FCN26:	LDA FRZDMA RAR JNC FCN26A LHLD USERDE SHLD USRDMA JMP FCN26B	; SET DMA ADDR. ALLOWED TO? ; IF NOT, SKIP THIS. ; REMEMBER WHAT USER WANTED	3 C0B3 C3 09 3C
FCN26A:	CALL DEFDMA	; MAKE IT SYSTEM STANDARD	
FCN26B:	JMP MRETN		
FCN27:	LHLD CURALV SHLD ANSWER JMP MRETN	; GET CURRENT DRIVE'S ALLOCATION VECTOR	3 C0F 22 0C 33
FCN28:	CALL MAKERO JMP MRETN	; UNDOCUMENTED. SET R/O BIT FOR LGD DISK.	
FCN29:	LDA RCVEC STA ANSARL JMP MRETN	; UNDOCUMENTED. GET R/O VECTOR	
; FREEZE DMA ADDR. DON'T LET USER CHANGE IT.			
FCN30:	LXI H, FRZDMA MVI H, \$01 CALL DEFDMA JMP MRETN	; UNDOCUMENTED ; AND FREEZE IT AT SYS STANDARD	3 C27 30 01
; DISPATCH TABLE FOR USER FUNCTIONS.			
FCNDTB:	• ADDR FCN0,FCN1,FCN2,FCN3,FCN4,FCNS • ADDR FCN6,FCN7,FCN8,FCN9,FCN10,FCN11 • ADDR FCN12,FCN13,FCN14,FCN15,FCN16,FCN17 • ADDR FCN18,FCN19,FCN20,FCN21,FCN22,FCN23 • ADDR FCN24,FCN25,FCN26,FCN27,FCN28,FCN29 • ADDR FCN30		

;COMMON RETURN TO USER

MRETN:	LDA	RELOGF	;NEED TO RE-LOG A DISK?	
	CPI	B		
	JZ	MRETN1	;JUMP IF NOT	
	LHLD	USERDE	;YES. POINT TO FCB	
	LDA	RELOGF	;PUT ENTRY TYPE BYTE BACK IN USER FCB	
	MOV M,	A		
	LDA	RELOGN	;HAVE TO RE-LOG THIS DISK	
	STA	ANSARG	;LOG FROM HERE	
	CALL	LGNDSK	;GO DO IT	
MRETN1:	LDA	ANSARL	;ANSWER TO USER	
	LXI D,	ANSWER		3 3C88 11 0C 33
	CALL	AORIDE	3 0A IOR (DE) => HL	
	XCHG			3 3C8E EB
	DCX H	H		3 3C8F 2B
	MOV M,	E		3 3C90 73
	INX H			3 3C91 23
	MOV M,	D		3 3C92 72
	RET			3 3C93 C9
	XCHG			3 3C94 EB
APLSMM:	MOV E,	A	3 0A + (HL) => HL	
	MVI D,	E		3 3C96 16 0E
	XCHG			3 3C98 EB
	LDAX	D		3 3C99 1A
	ADD	L		3 3C9A 85
	MOV L,	A		3 3C9B 6F
	INX	D		3 3C9C 13
	LDAX	D		3 3C9D 1A
	ADC	H		3 3C9E 80
	MOV H,	A		3 3C9F 67
	RET			3 3CA0 C9
AERHL:	MOV E,	A	3 0A IOR HL => HL	
	MVI D,	L		3 3CA2 16 0E
	MOV A,	E		3 3CA4 7B
	ORA	L		3 3CA5 B5
	MOV L,	A		3 3CA6 6F
	MOV A,	D		3 3CA7 7A
	ORA	H		3 3CA8 B4
	MOV H,	A		3 3CA9 67
	RET			3 3CAA C9
AORIDE:	XCHG		3 0A IOR (DE) => HL	
	MOV E,	A		3 3CAC 5F
	MVI D,	B		3 3CAD 16 0E
	XCHG			3 3CAF EB
	LDAX	D		3 3CB0 1A
	ORA	L		3 3CB1 B5
	MOV L,	A		3 3CB2 6F
	INX	D		3 3CB3 13
	LDAX	D		3 3CB4 1A
	ORA	H		3 3CB5 B4
	MOV H,	A		3 3CB6 67
	RET			3 3CB7 C9

;MULTIPLE SHIFT ROUTINE, DOING RLC'S

RLCNM: MOV A, M

3 3C88 7E

RLCN:	RLC		; 3CB9	07
DCR	C		; 3CBA	0D
JNZ	RLCN		; 3CBB	C2 B9 3C
RET			; 3CBE	C9
RALNM:	MOV A, M		; 3CBF	7E
RALN:	RAL		; 3CC0	0F
DCR	C		; 3CC1	0D
JNZ	RALN		; 3CC2	C2 C6 3C
RET			; 3CC5	C9
; PICK UP 16 BITS AT (HL), THEN DO (C) DAD H'S ON THEM				
SHLMM:	MOV E, M	PICK UP A DOUBLE BYTE		
	INX H			
	MOV D, M			
	XCHG	PUT THEM IN HL		
SHLMML:	DAD H	SHIFT THEM LEFT		
	DCR C			
	JNZ SHLMM	;(C) TIMES, AT LEAST ONCE		
	RET			
RARNM:	MOV A, M	PICK UP MEM ARG		
RARNL:	ORA A	CLEAR CARRY		
	RAR	ROTATE IT RIGHT		
	DCR C	THIS MANY TIMES		
	JNZ RARNL			
	RET			
MOV L, C		; 3CD8	69	
MOV H, B		; 3CD9	6A	
DEMIHL:	MOV C, M	16 BIT SUBTRACT 2 MEMS	TO HL	
	INX H		; 3CDB	23
	MOV B, M		; 3CDC	46
	LDAX D	;(DE) - (HL) => HL		
	SUB C		; 3CDE	91
	MOV L, A		; 3CDF	6F
	INX D		; 3CE0	13
	LDAX D		; 3CE1	1A
	SBB B		; 3CE2	98
	MOV H, A		; 3CE3	67
	RET		; 3CE4	C9
DEMIBA:	MOV L, A	; 0A => HL		
	MVI H, B		; 3CE6	26 00
DEMHL:	LDAX D	;(DE) - HL => HL		
	SUB L		; 3CE9	95
	MOV L, A		; 3CEA	0F
	INX D		; 3CEB	13
	LDAX D		; 3CEC	1A
	SBB H		; 3CED	9C
	MOV H, A		; 3CEE	07
	RET		; 3CEF	C9
^L				
; VARIABLES FOR BDOS PORTION				
ANSARL:	BYTE 0	RETURN USER'S VALUE HERE, FOR A REG		
ANSARG:	BYTE 0		; 3CF1	00
CHROFS:	BYTE 0	CHARACTER OFFSET OF FCB IN DIR BUFFER		
DOSFCB:	BYTE 0		; 3CF3	00

DIRSEC:	•BYTE 0	3 3CF4 00
SYSDMA:	•ADDR DEFBFR	•OUR COPY OF CURRENT DMA ADDR
USRDMA:	•ADDR DEFBFR	•USER'S REQUESTED DMA ADDR
FRZDMA:	•BYTE 0	
ALLOCS:	•BLKB NDISKS*NALLOV	•ALL THE ALLOCATION VECTORS
CURALV:	•BLKW 1	•CURRENT DISK'S ALLOC VECTOR
ROVEC:	•BYTE 0	•BITS FOR DRIVES THAT ARE READ-ONLY
CKSTAB:	•BLKB \$48	•NDISKS * NDIRECTORY-SEGMENTS
VECCKS:	•BLKW 1	•POINTER TO CHECKSUM TABLE
RELOGN:	•BYTE 0	•MRETN MUST RE-LOG THIS DISK
RFLGOF:	•BYTE 0	•FLAG NEED TO RE-LOG AT MRETN
LOGVEC:	•BYTE 0	•BITS FOR CURRENTLY LOGGED IN DISKS •B0=A, B1=B, B2=C, B3=D
TKPTRS:	•BLKB NDISKS	•CURRENT TRACK, FOR EACH DISK
TSECOV:	•BLKW NDISKS	•FOR ALLOCATING TRACKS
TKPNTR:	•BLKW 1	•POINTER TO TRACK NUMBER FOR CURRENT OP
TKSECO:	•BLKW 1	•POINTER INTO TRACKV, FOR ALLOC TRACK
CURRRC:	•BYTE 0	•RECORD COUNT
CURRNR:	•BYTE 0	•RECORD NUMBER
CURRMW:	•BYTE 0,0	•CURRENT MAP INFO, W/ 8 HIGH ORDER 0'S

•BELOW HERE ARE LOCAL SUBR ARG STORAGE

CTKTMP:	•BLKW 1	•TEMP IN ALLOCATION COMPUTATION
RNARG:	•BYTE 0	
RWCOMT:	•BYTE 0	
CKSTM1:	•BYTE 0	
CKSTM2:	•BYTE 0	
CRCURT:	•BYTE 0	
VERDIT:	•BYTE 0	
MAPNXV:	•BYTE 0	
GETBTW:	•BYTE 0	
MRKBTW:	•BYTE 0	
MRKBTW:	•BYTE 0	
ALOFCV:	•BYTE 0	
ALOFCW:	•BYTE 0	
ALOFCX:	•BYTE 0	
RDALCV:	•BYTE 0	
STFFIV:	•BYTE 0	
STPFIW:	•BYTE 0,0	
STPFIX:	•BYTE 0	
STPFIY:	•BYTE 0	
LOCKUV:	•BYTE 0,0,0,0	
ALO.LO:	•BYTE 0	
ALO.HI:	•BYTE 0	
CPYNAY:	•BYTE 0	
CPYNAW:	•BYTE 0	
CPNFIV:	•BYTE 0	
MAKFIV:	•BYTE 0	
MAKFIW:	•BYTE 0,0	
STPXTV:	•BYTE 0	
LTNXTV:	•BYTE 0	
WTNXTW:	•BYTE 0	

•A LITTLE SPACE TO END OF PAGE
•BYTE 0,0,0,0,0,0

TL

•HERE STARTS THE CUSTOMIZED BIOS

1

;FIRST THE VECTOR FOR DEFINED ENTRIES

BIOS:	JMP	0	;COLD BOOT
CBWBUT:	JMP	0	;WARM BOOT
CBTTCK:	JMP	0	;CONSOLE STATUS CHECK
CBCTYI:	JMP	0	;CONSOLE INPUT
CBCTYO:	JMP	0	;CONSOLE OUTPUT
CBLPTO:	JMP	0	;LIST OUTPUT
CBPTPO:	JMP	0	;PUNCH OUTPUT
CBPTRI:	JMP	0	;READER INPUT
CBHOME:	JMP	0	;HOME THE SELECTED DISK
CBSELE:	JMP	0	;SELECT A DISK
CBSTTK:	JMP	0	;SET TRACK
CBSTSC:	JMP	0	;SET SECTOR
CGSTMA:	JMP	0	;SET DMA ADDRESS
CBRFAD:	JMP	0	;READ A SECTOR
CBWRIT:	JMP	0	;WRITE A SECTOR

;REST NOT IN THIS SOURCE FILE

•END

TL
R