NOTICE

Corvus Systems, Inc. reserves the right to make improvements in the product described in this manual at any time and without notice.

DISCLAIMER OF ALL WARRANTIES AND LIABILITY

Corvus Systems, Inc. makes no warranties, either express or implied, with respect to this manual or with respect to the software described in this manual, its quality, performance, merchantability, or fitness for any particular purpose. Corvus Systems, Inc. software is sold or licensed "as is." The entire risk as to its quality and performance is with the buyer. Should the programs prove defective following their purchase, the buyer (and not Corvus Systems, Inc., its distributor, or its retailer) assumes the entire cost of all necessary servicing, repair, or correction and any incidental or consequential damages. In no event will Corvus Systems, Inc. be liable for direct, indirect, incidental, or consequential damages resulting from any defect in the software, even if Corvus Systems, Inc. has been advised of the possibility of such damages. Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you.

This manual is copyrighted and contains proprietary information. All rights are reserved. This document may not, in whole or part, be copied, photocopied, reproduced, translated or reduced to any electronic medium or machine readable form without prior consent, in writing, from Corvus Systems, Inc.

© 1980 by Corvus Systems, Inc.
2029 O'Toole Ave.
San Jose, CA 95131
(408) 946-7700

All rights reserved.

Reorder CORVUS product no. 11SM

Printed in U.S.A.
IMPORTANT!

Please Read Before Operating.

HANDLING INSTRUCTIONS:

CAUTION:  1. The CORVUS 11A is provided with a carriage lock to protect the disc surfaces during shipment. ALWAYS UNLOCK THE CARRIAGE LOCK PRIOR TO APPLYING POWER TO THE UNIT. ALWAYS LOCK THE CARRIAGE LOCK PRIOR TO MOVING OR SHIPPING THE UNIT.

2. The CORVUS 11A should be protected from undue shock and vibration. During shipment, the unit should be packaged in its original shipping container (or equivalent) unless the equipment in which it is installed is shipped in a manner which provides the necessary shipping protection.

The CORVUS 11A may be operated horizontally (on its base) or vertically on either side. It is not to be operated upside down or on end.

The CORVUS 11A draws cooling air through the bottom at the front. This provides adequate air flow to the drive. Insure that air flow is not restricted.

OPERATION OF CARRIAGE LOCK:

The surfaces of the discs can be damaged if the heads are allowed to slide across the surface when the drive is not in operation.

To prevent this occurring during shipping or handling, the drive is equipped with a carriage lock. This lock is located approximately in the center of the right (viewed from the front) side of the unit.

The carriage is unlocked by turning the screw in a clockwise direction for approximately 19 turns, or until resistance is encountered. Do not overturn as it is possible to damage the lock by applying too much force.

To lock the carriage, first insure that the carriage is fully retracted; that is, with the heads at track 0 or toward the outer edge of the discs. Turn the screw counter-clockwise approximately 19 turns or until it stops. Do not attempt to overturn by applying too much force.

With the assistance of a light it is possible to observe the movement of the lock. Its position can be determined by observing the location of the slot in the lock relative to its mounting pin. In either its locked or unlocked position, the pin will be at or near the end of the slot.
I. INSTALLATION

A. Unpacking

Your CORVUS 11S System includes the following:

1. The disk drive
2. The power supply
3. The CORVUS Personality Card
4. A DC power supply cable
5. An AC power supply cord
6. The CORVUS 11S
7. The CORVUS 11S disk handling and carriage lock instructions
8. CORVUS 11S Operating Instructions

NOTE: Please be sure to read the enclosed disk handling and carriage lock instructions before proceeding.

B. Cabling Instructions

You will find a flat cable exiting from the back of the CORVUS Disk Drive. This cable must be connected to the CORVUS Personality Card before the card is plugged into the S-100 Computer System.

The connector at the end of the cable should be attached to the set of pins on the CORVUS Personality Card. When the cable is connected correctly, the red stripe on the cable should be on the LEFT, and the cable should exit upwards away from its connector away from the Personality Card. Be sure that all the pins on the Personality Card’s connector go into the matching holes on the cable’s connector.

C. Installing the CORVUS Personality Card

1. TURN OFF THE POWER SWITCH AT THE BACK OF THE S-100 COMPUTER SYSTEM BEFORE PLUGGING IN THE CARD.
2. Remove the cover from the computer.
3. The CORVUS Personality Card may be plugged into any slot in the computer.
4. Insert the PCB edge connector into the chosen slot in the computer.
5. Replace the cover.

D. Connecting the Power Supply

The power supply must be connected to the disk via the DC power supply cable and to an AC outlet via the AC power supply cord.

The large square connector on the DC cable should be plugged into the power supply. You will notice that the connector has three squared
prompts which prohibit improperly attaching it to the power supply. When lined up properly, the connector should snap securely into the power supply.

The connector at the other end of the DC cable should be attached to the ten large pins at the upper right side of the rear of the disk drive. There should be a small red connector already attached to the two rightmost pins -- the DC connector should be on the ten leftmost pins. The cable should exit away from its connector on the side of the connector that is away from the disk drive.

The AC cord should be plugged into the power supply and into an AC outlet.

II. INTRODUCTION TO THE CORVUS 11S

The CORVUS 11S is an intelligent peripheral that adds cost effective mass storage to the S-100 Computer System’s Operating CP/M, while maintaining compatibility with existing hardware and software. The system package consists of the IHI 7710 “Winchester” disk drive with CORVUS Intelligent Controller, a complete Power Supply, and an Intelligent Module for the S-100, consisting of an interface card and its associated software.

III. DESCRIPTION OF THE CORVUS 11S

A. Ultra Compact 10 Megabyte Disk Drive

The disk drive is a technology leader that provides eleven million bytes of unformatted magnetic storage in less than two-thirds of a cubic foot of space. The unit features a closed loop servo. This assures accurate and rapid read/write head positioning independent of temperature and other environmental factors. There are three data surfaces and one servo on two eight inch platters.

The drive electronics are contained in three 7.5 inch by 10.5 inch printed circuit boards which are enclosed within the drive housing. This housing also contains a fourth PC card of the same dimensions which is the CORVUS Intelligent Disk Controller.

B. CORVUS Intelligent Controller

This controller is based on the 2-80 processor with 16K of Random Access Memory. Firmware for this controller provides such features as:

- Sector Buffering
- Read after Write
- Error recovery with automatic retries
- Transparent formatting with CRC error detection
- High speed data transfer utilizing DMA
INITIAL SYSTEM CHECKOUT

A. Verify Head Movement

This test insures that the head carriage has been released and is free to move.

1. Follow the enclosed "carriage lock instructions" for unlocking the head carriage.
2. Place the drive on a flat horizontal surface with rubber feet down and the power supply disconnected (or at least turned off).
3. Alternately lift one end (of the drive) and then the other and observe the head assembly move back and forth across the disc platter surface (about 3 CM of travel). If this does not occur, the carriage lock may still be engaged.

B. Verify Power Supply Operation

This test tests the power supply independently of the disc drive.

1. Plug the AC cord into the power supply and disconnect the DC power supply cable from the Corvus Drive.
2. Plug the AC cord into an AC outlet.
3. Toggle the "power switch" on the power supply. The switch should light up when the power supply is on. If this does not occur, the fuse may be damaged (or the AC outlet is not connected).
4. If you have a voltmeter, you may wish to test the actual voltages supplied by the unit. The voltages may be a little higher than the nominal values shown below because the supply is not being loaded down by the Corvus Drive.

VOLTAGES ON DRIVE/DC POWER PLUG
(TOP VIEW OF PLUG)

+5  +5  G  G  +12 -12 -5  G  +24
-------------
[ * ] [ * ] [ * ] [ * ] [ * ] [ * ] [ * ] [ * ]
-------------
[ ] [ ] [ ] [ ] [ ] [ ] [ ] [ ]

(WIRES)

NOTE: ALL VOLTAGES ARE MEASURED WITH RESPECT TO ONE OF THE GROUND LINES (G). THE ONLY WELL REGULATED VOLTAGES ARE THE +5V, -12V, AND -5V LINES. THE DRIVE DOES NOT WORK CORRECTLY IF THE VOLTAGES ARE OFF BY MORE THAN ABOUT 5% (WHEN LOADED BY THE DRIVE).

C. Drive Spin up Test

This test gives some indication that the power supply-
Disc drive combination are performing normally.

1. Place the drive on a flat horizontal surface with rubber feet down.
2. Connect the DC power cable to the drive and power supply and connect the AC power cord to an AC outlet and to the power supply.
3. Lift up the end of the drive where the cables are connected until the head assembly slides in toward the disc platter hub. Lower this end slowly so that the head remains near the hub (you can slide a book or magazine under this end to guarantee this).
4. Turn on the power supply. The disc platter should begin to spin up. When the drive comes up to the desired speed, the head should retract to the outer rim of the disc platter. If all this happens, you can proceed to the next test. If the drive does not spin up or the head does not retract, there is some hardware problem.

D. Drive Interface Test

This test checks some aspects of the combined Corvus drive, power supply, and computer interface. It assumes the use of some of the CP/M utilities supplied with the Corvus drive.

1. Follow the installation instructions supplied with the interface card (or "personality card") to connect your computer to the Corvus drive. Be sure that both the computer and the Corvus drive are turned off before this installation.
2. Connect up the power supply to the Corvus drive as in test C above.
3. Turn on the computer and then the drive power.
4. Boot in some CP/M system (not necessarily configured for the Corvus drive) and load the program CDIAGNOS.COM from the Corvus CP/M Interface/Utilities discette.
5. Select menu option #4 (head servo test) on drive 1. This should cause the head assembly (on the Corvus drive) to shoot back and forth across the disc. If this works, the system is probably working correctly. If the program hangs up after receiving the Corvus drive #, there is something wrong with the system (such as the drive is not up to speed yet, the interface is not installed properly, ...).

If the Corvus system is performing properly, you can proceed to the task of "personalizing" your CP/M for the Corvus drive. The programs:

- CLINK.ASM, PATCH.ASM for CP/M V 1.4X
- CLINK2.ASM for CP/M V 2.XX

may be the simplest method to interface your system to the Corvus Drive.
1. INDEX.DOC
   THIS DISC INDEX DOCUMENT FILE.

2. CORVUS.DOC
   A DOCUMENT FILE DESCRIBING HOW TO BRING UP CP/M 2.0
   ON THE CORVUS DRIVE.

3. UPDATE.DOC
   A DOCUMENT FILE DESCRIBING WHAT IS INVOLVED IN
   UPDATING A CORVUS DRIVE FROM VERSION 0 OF THE
   CONTROLLER CODE (USED ON ALL S-100 SYSTEMS SHIPPED
   BEFORE 2/26/80).

4. CERROR.DOC
   THIS IS A SHORT DOCUMENT FILE LISTING THE CONTROLLER
   ERROR CODES.

5. PATCH.ASM
   THIS IS AN OVERLAY PATCH FOR YOUR FLOPPY BASED CP/M 1.4x
   THAT ALLOWS THE CP/M TO ACCESS MORE THAN 4 DRIVES
   AND A PATCH TO ALLOW THE DRIVES TO BE LARGER THAN
   COMMON FLOPPY DRIVES. IT SHOULD NOT INTERFER WITH
   THE NORMAL OPERATION OF YOUR FLOPPY BASED CP/M 1.4x
   WITH THE POSSIBLE EXCEPTION OF A CASE WHERE YOU ATTEMPT
   TO ACCESS MORE THAN 4 DRIVES. THIS PROGRAM ONLY
   REQUIRES ABOUT 200H BYTES OF THE USER BIOS AREA, SO IT
   MAY BE SIMPLY CONFIGURED INTO HOST CP/M INTERFACES BY
   OVERLAYING THEM WITH THE PATCH.HEX FILE (IN THE
   USUAL CP/M CONFIGURATION PROCESS USING SYSGEN AND DDT).

6. CLINK.ASM
   THIS PROGRAM CAN BE USED IN CONJUNCTION WITH FLOPPY
   BASED CP/M 1.4x SYSTEMS THAT HAVE BEEN PATCHED WITH
   THE PATCH.ASM PROGRAM ABOVE. THIS PROGRAM LINKS THESE
   FLOPPY CP/M SYSTEMS TO THE CORVUS DRIVE BY INTERCEPTING
   VARIOUS BIOS DISC CALLS. TO USE IT, CREATE A PATCHED
   CP/M 1.4 SYSTEM WITH ABOUT 200H EXTRA BYTES OF RAM
   SOMEWHERE ABOVE IT. THEN USE THE EDITOR TO SELECT
   THIS BUFFER LOCATION IN THE CLINK.ASM PROGRAM (THE LABEL
   "FREE"). THEN PRODUCE A COM FILE FROM THIS SOURCE
   (CLINK.COM). YOU CAN NOW TRY IT OUT BY BOOTING UP
   YOUR PATCHED FLOPPY CP/M AND RUNNING CLINK.
   NOW TRY SELECTING DRIVES C,D,E,F,...,N. YOU SHOULD
   BE ABLE TO SEE AND/OR HEAR THE CORVUS HEAD MOVE
   (PARTICULARLY IF YOU SELECT DRIVE N FIRST, THEN C).
   WHEN YOU DO THIS THE FIRST TIME, THE PSEUDO DRIVES
   WILL ALL HAVE RANDOM DATA IN THEIR DIRECTORIES THAT
   WILL HAVE TO BE CLEANED UP A BIT. YOU CAN DO THIS
   WITH AN ERA ** COMMAND ON EACH OF THE PSEUDO DRIVES.

7. WHERE.ASM
   A SHORT PROGRAM USED WITH PATCH.ASM TO DETERMINE THE
   LOCATIONS OF VARIOUS CP/M ADDRESSES.
8. DIR.SUB
This is a submit file used for searching the directories of the pseudo drives setup in the CP/M 1.4 interface program: CLINK.ASM.

9. PUTGET.COM, PUTGET.ASM
This is a nice disc utility that can be used under CP/M to read and write from memory to the Corvus drive as well as fill various sections of the disc with data. The routine has its own disc drivers and is mainly useful as a SYSCON routine to write a configured CP/M 2.0 system out to the drive.

10. CLOADR.COM, CLOADR.ASM
This is a short boot loader program to be used with CP/M 2.0. It can be used under a floppy based CP/M to boot in CP/M from the hard disc (once it is put there) or it can be used to make a ROM based loader.

11. CBOOT.ASM
This is a cold boot loader for CP/M 2.0. It is brought in by CLOADR. CBOOT then brings in the CP/M system.

12. BIOSC.ASM
This is the source for the Corvus basic I/O system (BIOS) to configure into a copy of CP/M 2.0. This is initially setup to control four drives:

DRIVE A & B: Two pseudo drives on one Corvus drive. Each pseudo drive can hold about 4.85MB bytes.

DRIVE C & D: Two standard 8 inch single density soft sectored discs (in the standard CP/M format).

13. BIOSCT.ASM
This is a version of BIOSC.ASM with drivers for a Tarrell single density floppy disc controller.

14. CLINK2.ASM
This is a version of BIOSC.ASM that does not require any modification to your present floppy based CP/M 2.0 (2.1, 2.2, ...) except for possibly creating a 1K smaller system. It works by copying a set of Corvus disc drivers up above your present CP/M V2.0 system and linking them in to it. See file CORVUS.DOC for more information. This routine will probably be the simplest to use with any floppy based CP/M V2.0. However, the Corvus interface provided by BIOSC.ASM is much better because it warms boots off the hard disc.

15. CDIAGNOS.COM, CDIAGNOS.ASM
A "safe" Corvus disc diagnostic that can: read the controller code version #, check and correct disc format errors, and exercise the head (head servo test).

16. CREFORM.COM, CREFORM.ASM
THIS PROGRAM IS ONLY OF USE WHEN UPDATING FROM VERSION 0 CONTROLLER CODE (ALL S-100 SYSTEMS SHIPPED PRIOR TO 2/26/80 HAVE VERSION 0 CONTROLLER CODE). THIS PROGRAM PERMUTES THE DATA AND PROGRAMS ON THE HARD DISC TO A FORM COMPATIBLE WITH VERSION 1 (OR LATER) OF THE CONTROLLER CODE. THIS UPDATE IS REQUIRED FOR OPERATION WITH NEW CORVUS PRODUCTS SUCH AS "THE MIRROR".

17. CCODE.COM, CCODE.ASM
   THIS PROGRAM IS USED TO CHANGE THE CONTROLLER CODE OF A CORVUS DRIVE (THE CONTROLLER CODE ACTUALLY RESIDES ON PROTECTED TRACKS OF THE DRIVE AND IS BOOTED INTO RAM WHEN THE DRIVE SPINS UP).

18. CORVO.CLR
   THIS IS A CONTROLLER CODE FILE FOR USE WITH CCODE.COM.
   THIS IS A COPY OF THE ORIGINAL VERSION 0 CONTROLLER CODE.

19. CORV2.CLR
   THIS IS A CONTROLLER CODE FILE FOR USE WITH CCODE.COM.
   THIS IS A COPY OF THE VERSION 2 CONTROLLER CODE.

20. MIRROR.COM, MIRROR.ASM
   THIS PROGRAM IS THE CONTROL PROGRAM FOR THE CORVUS "MIRROR" DISC BACKUP SYSTEM. IT WILL NOT WORK UNDER VERSION 0 OF THE CONTROLLER CODE.

NOTE: THE SOURCES OF ALL PROGRAMS (ACCEPT FOR THE ACTUAL CONTROLLER CODE) ARE GIVEN BECAUSE:

1. THEY OFTEN CONTAIN AN EXPLANATION OF HOW TO USE THE PROGRAMS.
2. YOU MAY NEED TO CHANGE THE DISC I/O PORT ADDRESSES IF YOU HAVE A NON-STANDARD DRIVE INTERFACE.
3. WE ARE NOT TRYING TO KEEP ANY BIG SECRETS FROM YOU.

THE USE OF MOST OF THE UTILITY PROGRAMS IS EITHER FAIRLY OBVIOUS FROM THE PROMPTS, BY READING THE DOCUMENTATION OR BY SELF CONTAINED INSTRUCTIONS LISTED BY THE UTILITIES.

NOTE: ALL OF THE DISC UTILITIES CONTAIN THEIR OWN CORVUS DISC DRIVERS. THEY CAN BE RUN FROM FLOPPY BASED CP/M SYSTEMS THAT ARE NOT YET LINKED TO THE CORVUS DRIVE!!
THIS FILE DOCUMENTS HOW TO BRING CP/M V 2.0 UP ON THE CORVUS DRIVE.

THIS DISC SHOULD CONTAIN SEVERAL PROGRAMS TO ASSIST IN THIS TASK. THESE PROGRAMS ARE:

1. PUTGET.COM
   A NICE UTILITY THAT CAN BE RUN UNDER CP/M TO READ AND AND WRITE FROM MEMORY TO THE CORVUS DRIVE.

2. PUTGET.ASM
   THE COMMENTED SOURCE OF PUTGET.COM. IT ALSO HAS SOME INSTRUCTIONS ON ITS USE.

3. CLOADR.COM
   A SHORT BOOT LOADER PROGRAM THAT CAN BE LOADED UNDER CP/M (SAY FROM YOUR USUAL FLOPPY BASED SYSTEM) THAT WILL BOOT IN CP/M FROM THE CORVUS DRIVE (AFTER YOU PUT IT THERE).

4. CLOADR.ASM
   THE COMMENTED SOURCE OF CLOADR.COM. YOU MAY WISH TO USE THIS CODE TO MAKE A BOOT FROM SO THAT YOU CAN BOOT UP DIRECTLY ON THE CORVUS DRIVE.

5. CBIGHT.COM
   THE SOURCE OF A COLD BOOT LOADER THAT BOOTS IN CP/M FROM THE CORVUS DRIVE. THIS PROGRAM IS LOADED BY THE PROGRAM "CLOADR". THIS PROGRAM MUST BE CHANGED WHEN YOU CHANGE THE SIZE OF CP/M.

6. BIOSC.ASM
   THE SOURCE OF THE CORVUS BASIC I/O SYSTEM TO CONFIGURE INTO YOUR COPY OF CP/M V 2.0. THIS IS INITIALLY SETUP TO CONTROL FOUR DRIVES:
   DRIVE A & B: TWO PSEUDO DRIVES ON THE ONE CORVUS DRIVE. EACH PSEUDO DRIVE CAN HOLD ABOUT 4.85MBYTES.
   DRIVE C & D: TWO STANDARD 8 INCH SINGLE DENSITY SECTORED DISCS (IN THE STANDARD CP/M FORMAT).

7. BIOSCT.ASM
   A VERSION OF BIOSC.ASM WITH FLOPPY DISC I/O FOR A TARBEILL SINGLE DENSITY CONTROLLER.

8. CLINK2.ASM
   A VERSION OF BIOSC.ASM THAT DOES NOT REQUIRE ANY MODIFICATION TO YOUR PRESENT FLOPPY BASED CP/M 2.0 - EXCEPT FOR POSSIBLY CREATING A 1K SMALLER SYSTEM. IT WORKS BY COPYING A SET OF CORVUS DISC DRIVERS UP ABOVE YOUR PRESENT SYSTEM AND LINKING THEM INTO IT. THE CORVUS DRIVERS ARE INITIALLY SET UP TO ADDRESS THE CORVUS DISC AS TWO PSEUDO DRIVES IN THE SAME FORMAT USED IN BIOSC.ASM:
   DRIVE A & B: TWO FLOPPY DRIVES (OF ANY TYPE OR SIZE)
DRIVE C & D: TWO PSEUDO DRIVES ON THE ONE CORVUS DRIVE.
NOTE: THE CONTROL OF DRIVES A & B ARE ASSUMED TO BE SUPPLIED
BY YOUR CURRENT FLOPPY BASED CP/M 2.0 (2.1, 2.2, ...).

TO USE THIS PROGRAM:
1. CREATE A FLOPPY BASED CP/M WITH AT LEAST 350H BYTES
   OF EXTRA RAM AREA ABOVE THE PARTS OF YOUR FLOPPY
   BASED CP/M 2.X (ABOVE ANY BUFFERS OR TABLES USED
   BY THE BIOS ALSO).
2. EDIT A COPY OF CLINK2.ASM TO CHANGE THE EQUATE FOR
   THE LABEL: FREE TO POINT TO THE RAM AREA SELECTED
   IN STEP 1. IN MOST CASES THIS LOCATION CAN BE
   CHOSEN AS THE 1K AREA DIRECTLY ABOVE THE CURRENT
   CP/M.  THUS, FOR EXAMPLE A 63K CP/M WOULD ALLOW
   THE LAST 1K OF MEMORY TO BE USED FOR THE CORVUS
   DRIVERS.  IN THIS CASE WE WOULD CHOOSE:
   
   FREE EQU OFC00H

3. ASSEMBLE THIS FILE TO PRODUCE A COM FILE: CLINK2.COM.
4. CLEAN OUT THE DIRECTORY AREAS OF THE CORVUS DRIVE
   AS DESCRIBED BELOW IN STEP 2. OF THE EXAMPLE.
5. SPIN UP THE CORVUS DRIVE (IF NOT ALREADY TURNED ON).
6. BOOT UP UNDER THE CP/M CREATED IN STEP 1 (ABOVE) AND
   LINK IN THE CORVUS DRIVE BY RUNNING: CLINK2.COM
7. TRY SELECTING DRIVES C & D AND NOTICE THE HEAD MOVE
   ON THE CORVUS DRIVE.  IF THIS WORKS OK YOU CAN TRY
   COPYING SOME FILES TO THE HARD DISC, SAVING SOME TEST
   FILES WITH THE 'SAVE' COMMAND, AND OTHER TESTS.

9. CORVUS.DOC
   THIS DOCUMENT FILE.

--------------------------------------------------------------------------------

EXAMPLE: CONSTRUCT A 20K CP/M V 2.0 ON THE CORVUS DRIVE

THE EQUATES IN BIOSC AND CBOOT ARE NOW SETUP FOR A 20K CP/M
V 2.0, SO WE WILL USE THIS AS AN EXAMPLE.

1. THE FIRST TASK IS TO MODIFY BIOSC.ASM FOR YOUR CONSOLE AND
   OTHER I/O AS WELL AS TO ADD YOUR DISC DRIVERS FOR YOUR FLOPPYS.
   IT IS RECOMMENDED THAT YOU FIRST ONLY CHANGE THE CONSOLE I/O
   DRIVERS, THEN AFTER THIS WORKS YOU CAN ADD YOUR DISC DRIVERS.

   ONCE THAT YOU HAVE EDITED BIOSC.ASM, ASSEMBLE IT AND CBOOT.ASM
   TO PRODUCE TWO HEX FILES.  NOW YOU WILL NEED A COPY OF THE 20K
   CP/M V2.0.  USE SYSGEN TO GET IT OFF YOUR MASTER DISC THEN SAVE IT
   AS A CON FILE:

   A>SYSGEN
   SOURCE DRIVE (OR RETURN TO SKIP) B
   SOURCE ON B, THEN TYPE RETURN
   FUNCTION COMPLETE
   DESTINATION DRIVE NAME (OR RETURN TO REBOOT)

   A>SAYE 35 CPM20.COM
2. Now you must clean out the directory areas of the two pseudo drives (fill them with 0E5H). The directories are 64 sectors long (>one track) at disc address: 72 & 37944. To do this we will use the fill disc command in PUTGET:

A>PUTGET

--- CORVUS PUT/GET ROUTINE ---

( VERSION 1.2 )

PUT, GET, OR FILL (P/G/F) ? F
DRIVE # (1-4) ? 1
HEX BYTE TO FILL DISC WITH ? E5
STARTING DISC ADDRESS ? 72
NUMBER OF SECTORS ? 64
PUT, GET, OR FILL (P/G/F) ? F
DRIVE # (1-4) ? 1
HEX BYTE TO FILL DISC WITH ? E5
STARTING DISC ADDRESS ? 37944
NUMBER OF SECTORS ? 64
PUT, GET, OR FILL (P/G/F) ? C

A>

3. Now we can put in the CORVUS disc routines into CP/M and write it to the CORVUS disc. First use DDT to patch in CBOOT and BIOSC:

A>DDT
DDT VERS 1.4
-ICPM20.COM
-R
NEXT PC
2400 0100
-ICBOOT.HEX
-R900
NEXT PC
2400 0000
-IBIOSC.HEX
-RD580 <- OFFSET GIVEN BY VALUE OF "OFFSET"
NEXT PC
2400 0060
-L980,983 <- VERIFY CORRECT CP/M SIZE
0980 JMP 375C
0983 JMP -C
A>

NOTE: The offset: D580 assumes a "standard 20K CP/M 2.0" as described in the manuals from Digital Research. Some software houses and floppy disc system mfgs. ship a "personalized" version of CP/M that may actually be a 19K or 19.5K CP/M v2.0. In this case the value of the label: "DELTAS" in both CBOOT and BIOSC will have to be modified. In particular, suppose that the CP/M being used is actually a 19.5K CP/M. In this case, the value of the
JUMP ADDRESS AT THE BASE OF CCP EXAMINED ABOVE WOULD BE 355C, WHICH INDICATES THAT THE VALUE OF "DELTA" SHOULD BE CHANGED FROM 0000H TO 200H IN BOTH BIOSC.ASM AND CBOOT.ASM.

NOW USE PUTGET TO WRITE THIS ON THE DISC:

A>PUTGET

--- CORVUS PUT/GET ROUTINE ---

( VERSION 1.2 )

PUT, GET, OR FILL (P/G/F) ? P
DRIVE # (1-4) ? 1
STARTING HEX RAM ADDRESS ? 900
STARTING DISC ADDRESS ? 12
NUMBER OF SECTORS ? 60
PUT, GET, OR FILL (P/G/F) ? "C"

A>

4. NOW SEE IF IT WORKS BY BOOTING IN THE SYSTEM OFF THE CORVUS DRIVE WITH THE CLOADR PROGRAM:

A>CLOADR

---- CORVUS 20K CP/M V2.0 OF 2-26-80 ----

A>

5. YOU CAN NOW TRY SAVING SOME TEST FILES WITH THE SAVE COMMAND AND SEE IF THEY APPEAR IN THE DIRECTORY. ALSO YOU CAN TRY GIVING A WARM BOOT COMMAND WITH CONTROL-C. IF THIS WORKS OK, YOU CAN GO BACK AND ADD YOUR FLOPPY DISC DRIVERS TO BIOSC AND TEST IT OUT.

NOTE: IF YOUR DRIVERS MAKE THE CODE SECTORS OF BIOSC LONGER THAN 59, YOU WILL HAVE TO CHANGE THE LAYOUT OF THE PSEUDO DISCS SETUP IN BIOSC OR USE THE 12 RESERVED SECTORS (DISC ADDRESS: 0-11). YOU WILL ALSO HAVE TO CHANGE CBOOT AND POSSIBLY CLOADR.

SIMILARLY, THE COMBINATION OF THE TWO DISC DRIVERS IN THE BIOS MAY MAKE A COMBINED OPERATING SYSTEM LARGER THAN YOUR MEMORY ALLOWS (THE 20K CP/M MAY NOT FIT IN 20K OF MEMORY). IN THIS CASE, YOU MAY WISH TO USE A CP/M THAT IS 1K SMALLER THAN YOUR MEMORY SIZE.
THIS FILE DOCUMENTS THE DIFFERENCES AND INCOMPATIBILITIES BETWEEN SYSTEMS CONFIGURED WITH VERSION 0 OF THE DISC CONTROLLER CODE (SHIPPED ON ALL S-100 SYSTEMS PRIOR TO 2/26/80) AND LATER CONTROLLER CODE VERSIONS (VERS. 1 IN PARTICULAR). THIS FILE SHOULD MAINLY BE OF USE TO THOSE WHO ARE UPDATING FROM VERS. 0 CONTROLLER CODE TO USE NEW CORVUS PRODUCTS SUCH AS "THE MIRROR".

VERSION 1 OF THE CONTROLLER CODE WAS RELEASED WITH "THE MIRROR" AS THE FIRST "UNIVERSAL" VERSION THAT CAN BE USED ON ALL NON-DMA CORVUS INTERFACES (FOR APPLk, TRS-80, S-100, ALTOS, ...). IT INCLUDES COMMANDS FOR NEW CORVUS PRODUCTS SUCH AS "THE MIRROR" AS WELL AS THE ABILITY TO USE VARIABLE SECTOR SIZES (128, 256, AND 512 BYTE SECTORS). THIS CODE IS NOT DIRECTLY COMPATIBLE WITH PROGRAMS WRITTEN FOR OR DATA STORED ON THE DISC BY VERS. 0 OF THE CONTROLLER CODE. THE UPWARD INCOMPATIBILITIES ARE:

1. THE READ/WRITE COMMANDS FOR 128 BYTE SECTORS HAVE BEEN CHANGED FROM 2H/3H TO 12H/13H.

2. THE ORDER OF THE 128 BYTE SECTORS ON THE DRIVE HAS BEEN CHANGED (THE MIDDLE TWO SECTORS OUT OF EVERY FOUR HAVE BEEN PERMUTED) IN ORDER TO BE COMPATIBLE WITH THE FORMAT OF THE 256 AND 512 BYTE SECTORS. THE PROGRAM: CREFORM.COM HAS BEEN PROVIDED TO PERMUTE THE DATA ON DISCS WRITTEN WITH VERS. 0 CONTROLLER CODE TO THE VERS. 1 FORMAT.

IF YOU ARE NOW RUNNING UNDER VERS. 0 OF THE CONTROLLER CODE (YOU CAN FIND OUT WITH THE PROGRAM: CDIAGOS.COM) AND WISH TO UPDATE TO MORE RECENT CONTROLLER CODE VERSIONS, YOU HAVE SEVERAL CHOICES DEPENDING ON YOUR SITUATION. IN ALL CASES YOU WILL BE USING THE PROGRAM: CCODE.COM TO UPDATE YOUR CONTROLLER CODE (THE CONTROLLER CODE RESIDES ON PROTECTED TRACKS ON THE HARD DISC).

YOU CAN THEN USE THE PROGRAM: CREFORM.COM TO SWITCH THE DATA/PROGRAMS AROUND (PERMUTE THE SECTORS) ON YOUR DISC TO THE NEW FORMAT.

YOU MUST THEN RECONFIGURE YOUR CP/M DISC INTERFACE TO USE THE NEW READ/WRITE COMMANDS (12H/13H). IF POSSIBLE, YOU SHOULD USE THE NEW VERSIONS OF THE INTERFACE PROGRAMS PROVIDED WITH THIS UPDATE SINCE A FEW IMPROVEMENTS HAVE BEEN MADE IN THESE INTERFACE Routines.

IT IS PARTICULARLY IMPORTANT TO INSURE THAT THE VARIOUS PSEUDO DRIVES IMPLEMENTED ON THE SINGLE CORVUS DRIVE ALL START (THEIR DIRECTORY STARTS) ON A (128 BYTE) DISC ADDRESS (0 - 75743) THAT IS DIVISIBLE BY FOUR. THIS IS TO ALLOW "THE MIRROR" TO BACKUP ANY OF THESE PSEUDO DRIVES INDEPENDENTLY. IF YOU ARE USING OUR PROGRAM: CORVUS.ASM WITH THE ORIGINAL FORMAT PROVIDED, THERE IS NO PROBLEM. WE DID NOT CHANGE THIS FORMAT IN OUR NEW RELEASE. HOWEVER, WE DID HAVE TO CHANGE OUR CP/M 2.0 INTERFACE.

OUR ORIGINAL CP/M 2.0 INTERFACE HAD PSEUDO DRIVE A'S DIRECTORY STARTING AT A "GOOD" ADDRESS (#68 - WHICH IS DIVISIBLE BY 4).
UNFORTUNATELY, PSEUDO DRIVE B'S DIRECTORY FELL ACROSS A 512 BYTE BLOCK BOUNDARY. OUR NEW VERSION CORRECTS THIS BY CHANGING THE LOCATIONS OF BOTH DRIVES A & B. THERE IS NO PARTICULAR REASON TO ADOPT THIS NEW CONVENTION UNLESS YOU HAVE PURCHASED A "MIRROR". IF YOU HAVE, YOU MAY WISH TO SWITCH TO THE NEW FORMAT. THE FOLLOWING PROCEDURE ASSUMES THAT YOU WISH TO PRESERVE DATA/PROGRAMS THAT ARE ALREADY ON YOUR HARD DISC. TO UPDATE:

1. USE CCODE.COM AND CREFORM.COM TO CHANGE THE CONTROLLER CODE AND PERMUTE THE SECTORS.

2. RECONFIGURE YOUR OLD DISC INTERFACE Routines (BIOSC, CBOOT, AND CLOADR) TO USE THE NEW READ/WRITE COMMANDS. THIS REQUIRES CHANGING THE EQUATES FOR: RDCOM & WRCOM. THEN WRITE THIS NEW SYSTEM OUT TO THE DISC (IN THE WAY YOU DID BEFORE).

3. THIS SHOULD PUT YOU BACK ON THE AIR WITH THE NEW CONTROLLER CODE- BUT WITH A NON-OPTIMAL DISC ORGANIZATION (FOR "THE MIRROR").

4. USE "THE MIRROR" TO SAVE (BACKUP) PSEUDO DRIVE A ON VIDEO TAPE (STARTING BLOCK # 68/4=17, # BLOCKS=9440).

5. USE THE ERA *.* COMMAND (CP/M) TO CLEAR OUT DRIVE A AND THEN USE PIP TO COPY ALL FILES ON (PSEUDO) DRIVE B TO A.

6. USE "THE MIRROR" TO SAVE A COPY OF DRIVE A AGAIN.

7. NOW USE THE NEW VERSIONS OF THE DISC INTERFACE Routines (BIOSC, CBOOT, AND CLOADR INCLUDED IN THIS UPDATE) TO CONFIGURE A NEW CP/M SYSTEM ON THE DISC (IN THE NEW FORMAT).

8. USE "THE MIRROR" TO RESTORE THE COPIES OF DRIVES A & B TO THEIR NEW LOCATIONS:

   DRIVE A: STARTING BLOCK # = 18
   DRIVE B: STARTING BLOCK # = 9486

WELL THAT SHOULD DO IT. YOU EVEN GOT A TASTE OF HOW TO USE THE "MIRROR".
CORVUS DISC ERROR CODES

THE CORVUS CONTROLLER HAS A NUMBER OF ERROR CODES THAT MAY BE ISSUED IF EITHER AN ILLEGAL COMMAND IS GIVEN, OR THE CONTROLLER IS OUT OF SYNCHRONIZATION, OR THERE IS A HARDWARE MALFUNCTION. A NUMBER OF THE UTILITIES AND DISC INTERFACE PROGRAMS CAN LIST THESE ERROR CODES (IN HEX) IF SUCH AN ERROR OCCURS. FOR EXAMPLE, PUTGET.COM LIST THE CODE AS:

** DISC R/W ERROR # XXH **

WHERE XX IS THE ERROR CODE. YOU CAN DEMONSTRATE THIS BY TRYING TO READ A SECTOR (WITH PUTGET) FROM DRIVE 4 (UNLESS YOU HAVE FOUR DRIVES). THIS WILL GIVE ERROR CODE: A7H. THE UPPER 3 BITS OF THE ERROR CODE HAVE THE FOLLOWING SIGNIFICANCE:

BIT 5 : SET IF THERE WAS A RECOVERABLE ERROR (AS IN A RE-TRY ON READ OR WRITE).

BIT 6 : SET IF AN ERROR OCCURRED ON A RE-READ (VERIFICATION) FOLLOWING A DISC WRITE.

BIT 7 : SET IF ANY FATAL ERROR HAS OCCURRED.
NOTE: MOST OF THE PROGRAMS WILL NOT LIST THE ERROR UNLESS BIT 7 IS SET.

THE LOWER 5 BITS HAVE THE FOLLOWING SIGNIFICANCE:

<table>
<thead>
<tr>
<th>BITS 4-0</th>
<th>MEANING</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>DISC HEADER FAULT</td>
</tr>
<tr>
<td>1</td>
<td>SEEK TIMEOUT</td>
</tr>
<tr>
<td>2</td>
<td>SEEK FAULT</td>
</tr>
<tr>
<td>3</td>
<td>SEEK ERROR</td>
</tr>
<tr>
<td>4</td>
<td>HEADER CRC ERROR</td>
</tr>
<tr>
<td>5</td>
<td>RE-ZERO (HEAD) FAULT</td>
</tr>
<tr>
<td>6</td>
<td>RE-ZERO TIMEOUT</td>
</tr>
<tr>
<td>7</td>
<td>DRIVE NOT ON LINE</td>
</tr>
<tr>
<td>8</td>
<td>WRITE FAULT</td>
</tr>
<tr>
<td>9</td>
<td>READ DATA FAULT</td>
</tr>
<tr>
<td>A</td>
<td>DATA CRC ERROR</td>
</tr>
<tr>
<td>B</td>
<td>SECTOR LOCATE ERROR</td>
</tr>
<tr>
<td>C</td>
<td>WRITE PROTECTED</td>
</tr>
<tr>
<td>D</td>
<td>ILLEGAL SECTOR ADDRESS</td>
</tr>
<tr>
<td>E</td>
<td>ILLEGAL COMMAND</td>
</tr>
<tr>
<td>10</td>
<td>DRIVE NOT ACKNOWLEDGED</td>
</tr>
<tr>
<td>11</td>
<td>ACKNOWLEDGE STUCK ACTIVE</td>
</tr>
<tr>
<td>12</td>
<td>TIMEOUT</td>
</tr>
<tr>
<td>13</td>
<td>FAULT</td>
</tr>
<tr>
<td>14</td>
<td>CRC</td>
</tr>
<tr>
<td>15</td>
<td>SEEK</td>
</tr>
<tr>
<td>16</td>
<td>VERIFICATION</td>
</tr>
<tr>
<td>17</td>
<td>DRIVE SPEED ERROR</td>
</tr>
<tr>
<td>Code</td>
<td>Error Description</td>
</tr>
<tr>
<td>------</td>
<td>------------------------------------------</td>
</tr>
<tr>
<td>18</td>
<td>DRIVE ILLEGAL ADDRESS ERROR</td>
</tr>
<tr>
<td>19</td>
<td>DRIVE R/W FAULT ERROR</td>
</tr>
<tr>
<td>1A</td>
<td>DRIVE SERVO ERROR</td>
</tr>
<tr>
<td>1B</td>
<td>DRIVE GUARD BAND</td>
</tr>
<tr>
<td>1C</td>
<td>DRIVE FLO (PHASE LOCK) ERROR</td>
</tr>
<tr>
<td>1D</td>
<td>DRIVE R/W UNSAFE</td>
</tr>
</tbody>
</table>
CP/M 1.4 PATCH ROUTINE TO USE WITH CLINK.ASM
TO LINK A FLOPPY CP/M 1.4 TO THE CORVUS DRIVE

VERSION 1.02 FOR CP/M VERSION 1.4X

COPYRIGHT 1980 CORVUS. ALL RIGHTS RESERVED

--- THIS VERSION IS SET TO INSERT A SMALL AMOUNT OF CODE IN THE
BIOS AREA AT BIOS+5EOH (WHICH IS IN THE "USER" AREA OF
THE CONFIGURED CP/M SUPPLIED BY LIFEBOATS ASSOC.
FOR A DOUBLE DENSITY NORTH STAR DRIVE.
YOU SHOULD CHECK YOUR I/O ROUTINES IN THE USER I/O
AREA TO INSURE THAT THE SHORT ROUTINE IN THE USER
AREA "ELSWHR" DOES NOT OVERTAKE YOUR I/O ROUTINES

MSIZE EQU 22 ;CP/M SIZE IN KB
DELA EQU 000GH ;OFFSET FROM STD CP/M
BIAS EQU ((MSIZE-16)*1024-DELA) ;OFFSET FROM 16K CP/M
MAXDRV EQU 14 ;NUMBER OF CONFIGURED DRIVES
FREE EQU 3EOOH+5EOH+BIAS ;FREE AREA FOR EXTRA BIOS SPACE
 ;THE SHORT ROUTINE "ELSWHR" IS
 ;PUT HERE

STVEC EQU 349AH+BIAS ;ROUTINE TO SET VECTOR BIT
LGVEC EQU 3DC1H+BIAS ;LOGIN VECTOR LOCATION
ALLOC EQU 35EOH+BIAS ;ALLOC. MAP ROUTINE
DISKNO EQU 330BH+BIAS ;BUFFER FOR DRIVE #
SLERR EQU 310BH+BIAS ;SELECT DISK ERROR ROUTINE ADDRESS
ROTATE EQU 3CCH+BIAS ;ROTATE A BYTE ROUTINE
TRTAB EQU 3DC2H+BIAS ;LOGICAL TRACK TABLE
LCRACK EQU 3DC1H+BIAS ;POINTER FOR LOG. TRACK
SCTAB EQU 3DC6H+BIAS ;LOGICAL SECTOR TABLE
LSEC EQU 3DD0H+BIAS ;POINTER FOR LOG. SECTOR
SMTAB EQU 3DD7H+BIAS ;CHECKSUM TABLE
CKSUN EQU 3DD8H+BIAS ;POINTER TO CHECKSUM ARRAY
MPTAB EQU 3CF0H+BIAS ;ALLOCATION MAP TABLE
MAPP EQU 3D7AH+BIAS ;POINTER TO DRIVE ALLOC. MAP
ALTSER EQU 3A18H+BIAS ;ADDRESS OF DISC LOG-IN CODE
ILAGE EQU 310FH+BIAS ;INTERLACE ROUTINE ADDRESS

END OF IMPLEMENTATION DEPENDANT CODE.
CILACE EQU 33B9H+BIAS ;CALL TO INTERLACE ROUTINE
BTABLE EQU 31A9H+BIAS ;BDOS DRIVE INFO TABLE
SPACE EQU 3A93H+BIAS ;AREA TO PATCH TO GET SOME SPACE
SELDISK EQU 3E18H+BIAS ;BIOS SELECT DISK ROUTINE
SETSPL EQU 3E21H+BIAS ;BIOS SET SECTOR ROUTINE
;
OFFBIOS EQU 3600H+BIAS ;OFFSET TO BEGINNING OF BIOS
;

THIS ROUTINE PROCESSES DISK SELECTS INSIDE CP/M

ORG ALTSSEL ;THIS REPLACES NORMAL DISC LOGIN CODE

LDA DISKNO ;GET DRIVE #
CPI MAXDRV ;IS IT TOO BIG?
JC SELL ;IF DRIVE # IS VALID, DO SELECT
LHLD SLERR ;GET ADDRESS OF ERROR ROUTINE
PCHL ;ISSUE ERROR
;
SELL: LXI H,DTAB+3 ;POINT TO DRIVE TABLE
MVI C,1 ;SET SEARCH COUNT
SC1: CMP M ;TEST IF DRIVE IS LOGGED IN
JZ ELSWARN ;IF FOUND, SET POINTERS
DCX H ;OTHERWISE POINT TO NEXT LOCATION
DCR C ;COUNT DOWN TABLE SIZE (4 DRIVES)
JP SCI ;SEARCH THRU TABLE
;
SC2: LXI H,OPEN ;POINT TO OPEN COUNTER
PUSH PSW ;SAVE DRIVE #
MVI A,3 ;MASK FOR MOD 4 ARITH.
INR M ;INCREMENT COUNTER
ANA M ;MASK IT
MOV M,A ;SAVE BACK IN COUNTER
MOV C,A ;GET BYTE IN (B,C)
MVI B,0
LXI H,DTAB ;POINT TO DRIVE TABLE
DAD B ;INDEX INTO IT
POP PSW ;GET DRIVE # BACK
MOV M,A ;PUT DRIVE # IN TABLE
CALL SC3 ;SET POINTERS AND SELECT DRIVE
;
FIXIT: LDA LGVEC ;GET LOGIN VECTOR
MOV C,A
CALL STVEC ;SET LOGIN VECTOR
STA LGVEC
JMP ALLOC ;DEVELOPE ALLOC, MAP AND LOGON
;
SC3: MOV A,C ;GET COUNTER
STA MOD4 ;SAVE IT IN BUFFER
STPTR: MOV L,C ;GET MOD4 COUNTER
MVI N,D ;INTO (H,L)
XCHG ;SAVE IT IN (D,E)
LXI H,TRTB ;POINT TO LOG. TRACK TABLE
DAD   D                   ; INDEX TO TABLE LOCATION
SHLD  LGTRK              ; SAVE POINTER
XCHG
DAD   H                   ; DOUBLE IT
XCHG
LXI   H, SCTAB            ; POINT TO LOG. SECTOR TABLE
DAD   D                   ; INDEX INTO TABLE
SHLD  LGSEC              ; SAVE POINTER
XCHG
DAD   H                   ; GET COUNTER BACK
DAD   H
DAD   H                   ; COUNTER IS NOW 16 TIMES ORIG. VALUE
XCHG
LXI   H, SMTAB            ; POINT TO CHECKSUM TABLE (4 BYTES/DRIVE)
DAD   D                   ; INDEX INTO TABLE
SHLD  CKSUM               ; SAVE POINTER
XCHG
DAD   H                   ; PUT BACK IN (D,E)
LXI   D, MPTAB            ; POINT TO ALLOC. MAP TABLE
DAD   D                   ; INDEX INTO TABLE
SHLD  MAPP                ; SAVE POINTER TO DRIVE ALLOC. MAP
LDA   DISKNO              ; GET DRIVE #
MOV   C, A
JMP   SELDSK              ; NEED SOME EXTRA CODE AREA

OPEN: DB 3                 ; COUNTER/POINTER FOR OPEN TABLE POSITION

ORG  SPACE                ; RE-CODE TO FREE A FEW BYTES

RZ
STA   DISKNO
CALL   ALTSSEL
RET

; THAT LEAVES 5 BYTES FOR DISK SELECT TABLES

MOD4: DB 3                 ; POINTER TO OPEN TABLE POSITION
      ; INITIALIZE THE DRIVE TABLE

DTAB: DB OFFH, OFFH        ; INITIALIZE THE DRIVE TABLE
      DB OFFH, OFFH

PATCHES TO SOME OTHER ROUTINES IN CP/M

LDA   MOD4                ; GET INDEX TO DRIVE #

LDA   MOD4

LDA   MOD4

ORG   349EH+BIAS

ORG   3972H+BIAS

CALL   33F4H+BIAS         ; FIX BUG IN CP/M
NOP
NOP
NOP
MOV   A, C
LXI   H, 3DF8H+BIAS
ORG FREE

THE FOLLOWING CODE MUST BE PUT IN THE USER AREA
ABOVE BIOS (IN BIOS). IN THE CASE OF MANY SYSTEMS,
IT CAN BE OVERLayed ABOVE THE USUAL USER BIOS AREA
WITHOUT INTERFERING WITH THE CODE ALREADY IN USE.
HOWEVER, THE EXAMPLE BELOW ASSUMES THAT THE CODE
HAS BEEN PUT IN THE SHIFTED USER AREA PROVIDED
BY LIFEBOATS IN THEIR NORTH STAR CP/M 1.4A,
AND POSSIBLY OTHER CONFIGURATIONS.

--- CONTINUATION OF CP/M PATCH TO ALLOW MORE THAN
4 DRIVES. NOTE, THIS PATCH IS NOT COMPLETELY
FOOLPROOF. CP/M 1.4 CAN ONLY ADDRESS 4 DRIVES
AT A TIME. THIS PATCH JUST CONSTRUCTS A MAPPING
THAT MAPS THESE 4 LOGICAL DRIVES ONTO ANOTHER SET
OF 4 PHYSICAL DRIVES. THIS IS DONE BY A FIRST IN
FIRST OUT ALLOCATION SCHEME. IF YOU LOGON 4 DRIVES
AND THEN LOGON A NEW DRIVE, THE FIRST OF THE OLD 4
DRIVES WILL BE DE-ALLOCATED AND THE NEW DRIVE WILL
BE ASSIGNED TO THIS LOGICAL DRIVE #. THIS HAS
A SLIGHT POTENTIAL FOR ERROR IF YOU HAVE A PROGRAM
THAT IS READING AND WRITING TO MORE THAN 3 DRIVES AT
A TIME. IF IT SHOULD HAVE A DISC FILE OPENED FOR
WRITING AND HAVE THIS DRIVE BECOME DE-ALLOCATED
WHEN A NEW DRIVE IS LOGGED ON, DATA WILL BE LOST
BECAUSE THE MOST RECENT FILE CONTROL BLOCK FOR
THAT FILE PROBABLY HAD NOT BEEN WRITTEN OUT TO THE
DISC WHEN THE NEW DRIVE WAS LOGGED ON.

ELSWHR: CALL SC3
LDA MOD4 ;GET POINTER
INR A
MOV C,A ;SAVE FOR LATER
LDA LGVEC ;GET LOGON VECTOR
RCL CALL ROTATE ;ROTATE BYTE
RAR RC
JMP FIXIT ;OK IF ALREADY LOGGED IN
END
FILE: CLINK  ASM  PAGE 001

**********************************************************************
* *
* CORVUS FLOPPY CP/M LINK PROGRAM
* *
* FOR CP/M 1.4X
* *
* VERSION 1.05
* *
* FOR CP/M VERSION 1.4X
* *
* COPYRIGHT 1980 CORVUS.
* *
* ALL RIGHTS RESERVED
* *
**********************************************************************

THIS PROGRAM IS DESIGNED TO LINK A FLOPPY DISC CP/M V1.4X
THAT HAS BEEN PATCHED WITH THE FILE: PATCH.ASM TO THE
CORVUS HARD DISC. IF YOUR CP/M HAS TWO FLOPPYS AND
YOU CHOOSE TO KEEP THE CORVUS DISC LAYOUT CHOSEN BY THIS
PROGRAM, YOU NEED ONLY CHOOSE WHERE THIS PROGRAM IS TO
LOAD (SEE THE EQUIATE FOR FREE IN THIS PROGRAM).

THIS PROGRAM LOADS AT THE CP/M TPA (100H) AND MOVES A
PART OF ITSELF UP TO LOCATION:FREE AND THEN PATCHES
THE CODE IN THIS AREA TO LINK TO THE CURRENTLY RUNNING
FLOPPY CP/M 1.4X. THIS INVOLVES COPYING SOME TABLES
FROM THE CP/M UP INTO THE LINK CODE AND RE-DIRECTING SOME
JUMPS IN THE BIOS JUMP TABLE UP TO THE LINK CODE. THIS
LINK WILL NOT BE BROKEN BY A WARM BOOT BUT A COLD BOOT
WILL BRING IN A NEW COPY OF CP/M THAT IS NOT LINKED.
THIS PROGRAM MUST NOT BE RUN MORE THAN ONCE AFTER EACH
COLD START SINCE IT WOULD TRY TO SORT OF LINK TO ITSELF
IF RUN AGAIN (NOTE: THERE IS A BUILT IN CHECKING ROUTINE
TO PREVENT THIS LINK FROM BEING ENABLED TWICE- TO AVOID
THIS PROBLEM). THIS PROGRAM IS AN IDEAL CANDIDATE FOR USING
THE PROGRAM AUTO LOAD FEATURE BUILT INTO THE CCP OF
CP/M 1.4. THIS FEATURE ALLOWS CP/M TO AUTOMATICALLY
EXECUTE A PRE-SELECTED COMMAND (SUCH AS LOADING BASIC
OR DOING A DIRECTORY LISTING) ON EACH WARM OR COLD BOOT.
THIS FEATURE CAN BE MADE SELECTIVE TO ONLY COLD BOOTS
BY A MODIFICATION OF THE USER BIOS.

IN ANY CASE, YOU CAN ALWAYS DO THE LINK MANUALLY
BY LOADING CLINK AFTER YOU FIRST BOOT UP. NOTE, A CLEANER
SOLUTION IS TO CONFIGURE THESE DRIVERS DIRECTLY INTO YOUR
CP/M. HOWEVER THE CORVUS DRIVERS TAKE ABOUT 200H BYTES
OF CODE - WHICH YOU MAY NOT HAVE ROOM FOR IN THE REMAINING
BOOT TRACKS OF YOUR FLOPPY BASED CP/M.

THE DEFAULT FORMAT CHOSEN HERE IS:

DRIVE A & B : FLOPPY DISC DRIVES

DRIVE C - M : 512K BYTE PSEUDO DRIVES ON THE CORVUS DRIVE

DRIVE N : 3824K BYTE PSEUDO DRIVE ON THE CORVUS DRIVE

**********************************************************************
FMX EQU 2 ;NUMBER OF FLOPPY DRIVES
FREE EQU 00000H ;FREE AREA FOR THIS CODE TO
;BE MOVED TO ABOVE CP/M

*******************************************************************************

CP/M INTERNAL ADDRESSES
THESE ADDRESSES REFER TO A STANDARD 16K CP/M 1.4
THEY ARE USED AS OFFSETS TO LOCATE THE POSITIONS
IN THE CP/M THAT IS LOADED (NO MATTER WHAT ITS SIZE
IS). DO NOT CHANGE THEM FOR DIFFERENT CP/M SIZES.

ILACE EQU 310F0H ;INTERLACE ROUTINE ADDRESS
CILACE EQU 33B9H ;CALL TO INTERLACE ROUTINE
BTABLE EQU 313AH ;BDOS DRIVE INFO TABLE
CONOUT EQU 3E00H ;LOCATION OF CONOUT JUMP IN BIOS

--- CP/M FUNCTION EQUATES ---

BDOS EQU 05H ;LOCATION OF BDOS JUMP
LST EQU 9 ;LIST STRING COMMAND
RSET EQU 13 ;BDOS RESET COMMAND

I/O EQUATES FOR HARDWARE

DIDATA EQU 00EH ;DISK DATA PORT (R/W)
DISTAT EQU 0DFH ;STATUS PORT
DREADY EQU 1H ;READY LINE
DFACT EQU 2H ;NOT(If ACTIVE)
RDCH EQU 12H ;CORVUS READ COMMAND (VERS. 1 CCODE)
WRCH EQU 13H ;CORVUS WRITE COMMAND
DELAY EQU 50 ;APPROX DELAY TIME IN US

ORG 100H ;START AT CP/M TPA

--- COMPUTE ADDRESS OF BIOS FOR THE LOADED CP/M ---

LHLD 1 ;GET ADDRESS OF JUMP TO WBOOT
DCX H
DCX H
SHLD LDBIOS ;SAVE IT FOR LATER USE

--- COMPUTE CP/M OFFSET BIAS FROM 16K SYSTEM ----
MOV A,H
SUI 03EH
MOV H, A
SHLD CBIAS ; SAVE IT

; --- DETERMINE IF LINK IS ALREADY INSTALLED ---
LXI D, CONOUT+13 ; POINT TO 16K HOME ADDRESS
DAD D ; ADJUST FOR CP/M SIZE
LXI D, SHOME ; GET ADDRESS OF NEW VALUE
MOV A, M ; GET LOW ADDRESS BYTE
CMP E ; IS THERE A MATCH?
JNZ OK ; NO, SO LINK IS NOT INSTALLED
INX H ; POINT TO HIGH ADDRESS BYTE
MOV A, M ; GET IT
CMP D ; IS THERE A MATCH?
JNZ OK ; NO, SO LINK IS NOT INSTALLED

LXI D, LMSG ; POINT TO ERROR MESSAGE
MVI C, LIST ; SET FOR LIST FUNCTION
JMP BDSOS ; LIST AND EXIT BACK TO CP/M

; --- COPY CODE UP TO 'FREE' LOCATION -----
OK:
LXI H, START ; SOURCE OF CODE
LXI D, FREE ; DESTINATION OF CODE
LXI B, LENC+2 ; LENGTH OF CODE
CALL MOVE

; --- COPY PART OF OLD BIOS JUMP TABLE UP TO LINK PGM ---
LHLD CBIAS ; GET OFFSET
LXI D, 3E18H ; 16K ADDRESS OF PART OF BIOS TABLE
DAD D ; ADJUST FOR CURRENT CP/M SIZE
PUSH H ; SAVE IT
LXI D, FHOME ; DESTINATION
LXI B, 21
CALL MOVE
POP D

; --- COPY NEW LINK TABLE INTO BIOS JUMP TABLE ---
LXI H, NTAB ; NEW TABLE
LXI B, 21 ; SET TABLE LENGTH
CALL MOVE

; --- COPY FLOPPY CONFIGURATION TABLE UP INTO LINK PGM ---
LHLD CBIAS
LXI D, BTABLE ; LOCATION OF BDOS DRIVE INFO TABLE
DAD D ; CORRECT FOR CP/M SIZE
SHLD PTO+1 ; PATCH REFERENCE IN LINK PGM
SHLD PTOO+1
LXI D, FSIZE ; DESTINATION
LXI B, 7 ; SIZE OF INFO TABLE
CALL MOVE

; --- PATCH IN OTHER CP/M SIZE DEPENDENT ADDRESSES IN LINK PGM ---
FILE: CLINK  ASM  PAGE 004

; LHL D CBIAS
LXI D,CILACE ;CALL TO INTERLACE ROUTINE IN CP/M 1.4
DAD D ;ADJUST FOR CP/M SIZE
SHLD PTX1+1
SHLD PTX2+1
LHL C CBIAS
LXI D,ILACE ;INTERLACE ROUTINE ADDRESS
DAD D
SHLD PTX3+1

; --- NOTIFY OF CORVUS LINK ---

; LXI D,BMSG
MVI C,LST
CALL BDOS

; DO A SYSTEM RESET

; MVI C,RSET
JMP BDOS ;DO A RESET AND RE-ENTER CP/M (LINK IS DONE)

; MOVE:

; MOV A,M
STAX D
INX H
INX D
DCX B
MOV A,B
ORA C
JNZ MOVE
RET

; --- NEW JUMP TABLE TO BE COPIED INTO THE BIOS ---

; NTAB:
JMP SHOME ; JUMP TO SWITCH TABLE
JMP SELECT
JMP SETTRK
JMP SECSET
JMP SETDMA
JMP SREAD
JMP $WRITE

; BMSG:
DB ODH,OAH,"--- CORVUS LINK INSTALLED ---","ODH,OAH,"$
LMSC:
DB ODH,OAH,07,"** CORVUS LINK ALREADY INSTALLED **","ODH,OAH,"$
LDBIOS DS 2 ;BUFFER FOR BIOS LOCATION
CBIAS DS 2 ;BUFFER FOR CP/M BIAS

; START EQU $ ;START OF CODE TO BE MOVED UP

; SHIFT EQU FREE-START ;OFFSET OF CODE TO MOVE UP LOCATION

; NOTE: ALL LABELS IN THE CODE TO FOLLOW MUST BE
; OF THE FORM LABEL EQU $+SHIFT
; TO MAKE THE CODE CORRECTLY BE ASSEMBLED
; FOR THE SHIFTED ORIGIN (AT 'FREE').
; --- COPY OF ORIGINAL BIOS SELECT, SETTRK, SETSEC, AND SETDMA

FHOME EQU $+SHIFT
JMP 0
FSELEC EQU $+SHIFT
JMP 0 ;THIS GETS PATCHED ON STARTUP
FSTTRK EQU $+SHIFT
JMP 0
FSTSEC EQU $+SHIFT
JMP 0
FSTDMA EQU $+SHIFT
JMP 0
FREAD EQU $+SHIFT
JMP 0
FWRITE EQU $+SHIFT
JMP 0

; DS 2 ;EXTRA ROOM

; --- THIS JUMP TABLE IS USED AS A SWITCH TO DIRECT THE BIOS
; DISC INTERFACE CALLS TO THE FLOPPY OR HARD DISC ROUTINES.

SHOME EQU $+SHIFT
JMP FHOME ;SET TO FLOPPY ROUTINES AT FIRST
SREAD EQU $+SHIFT
JMP FREAD
SWRITE EQU $+SHIFT
JMP FWRITE

; DS 2 ;EXTRA ROOM

; --- THIS JUMP TABLE IS USED TO COPY INTO THE SWITCHING
; JUMP TABLE TO LINK TO THE FLOPPY DISC (WITH THE
; SELECT ROUTINE).

FTAB EQU $+SHIFT
JMP FHOME
JMP FREAD
JMP FWRITE

; DS 2 ;EXTRA ROOM

; --- THIS JUMP TABLE IS USED TO COPY INTO THE SWITCHING
; JUMP TABLE TO LINK TO THE HARD DISC (WITH THE
; SELECT ROUTINE).

HTAB EQU $+SHIFT
JMP HHOME
JMP HREAD
JMP HWRITE

; DS 2 ;EXTRA ROOM

; READ COMMAND

;
FILE: CLINK  ASM  PAGE 006

HREAD EQU $+SHIFT
CALL ADDRESS ;CALCULATE THE DISK ADDRESS

; NVI C,RDCOM ;GET READ COMMAND
; CALL WAITOUT ;WAIT AND OUTPUT WHEN READY
; LDA PDIVRE ;GET DRIVE $ IN C
ADD B ;ADD ADDRESS EXTENTION
MOV C,A ;EXTENDED DRIVE NUMBER IN C
CALL WAITOUT
; MOV C,L ;GET LOW ORDER ADDRESS
CALL WAITOUT
; MOV C,H ;GET HIGH ORDER ADDRESS
CALL WAITOUT
; COMMAND IS SET UP, NOW WAIT FOR RETURN
; CALL TURN ;TURN AROUND WAIT
; IN DIDATA ;GET RETURN CODE
ANI 80H ;SET FLAGS
JZ GETDATA ;IF POSITIVE THEN NO HARD ERROR
NVI A,1 ;ELSE ERROR CODE FOR BDOS
RET ;WITH ERROR
;
; STATUS WAS OK, SO NOW GET THE DATA
;
GETDATA EQU $+SHIFT
LHLD DMAADD ;GET DMA ADDRESS IN H,L
MVI C,128 ;WANT 128 BYTES OF DATA
;
; READ THE DATA
; DON'T USE CALL TO WAIT BECAUSE IT WILL SLOW THINGS UP.
;
; RWAIT EQU $+SHIFT
; IN DSTAT ;GET STATUS
ANI DREADY
JNZ RWAIT ;WAIT FOR READY
; IN DIDATA ;GET DATA
MOV M,A ;PUT IN MEMORY
INX H ;INCREMENT MEMORY POINTER
DCR C ;DONE YET?
JNZ RWAIT ;NO- GET MORE
; MVI A,0 ;DONE, AND NO ERRORS
RET ;END OF READ
;
;
WRITE COMMAND
FILE: CLINK  ASM  PAGE 007

HWRITE  EQU $+SHIFT
CALL  ADDRESS ;CALCULATE ADDRESS

MVI  C,WRCOM ;GET WRITE COMMAND
CALL  WAITHOLD ;

LDA  PDRIYE ;GET DRIVE #
ADD  B ;ADD ADDRESS EXTENTION
MOV  C,A ;EXTENDED DRIVE # IN C
CALL  WAITHOLD ;

MOV  C,L ;GET LOW ADDRESS
CALL  WAITHOLD ;

MOV  C,H ;GET HIGH ADDRESS
CALL  WAITHOLD ;

LHLD  DMAADD ;GET DMA ADDRESS IN H,L

WRITE COMMAND IS SET UP-NOW SEND DATA

WBLOCK  EQU $+SHIFT
MVI  C,128 ;RECORD LENGTH

WWAIT  EQU $+SHIFT
IN  DISTAT ;SAME OLD STUFF
ANI  DREADY
JNZ  WWAIT ;WAIT UNTIL READY

MOV  A,M ;GET DATA IN A
OUT  DIDATA ;SEND IT TO DRIVE
INX  H ;NEXT DATA BYTE
DCR  C ;END OF BLOCK?
JNZ  WWAIT ;NO, PUT SOME MORE

CALL  TURN ;TURN AROUND AND WAIT

IN  DIDATA ;GET RETURN CODE
ANI  80H ;SET FLAG BITS
MVI  A,1 ;ASSUME ERROR
RM
MVI  A,0 ;RETURN IF ERROR
REI

END OF WRITE

WAITHOLD ROUTINE
WAITS FOR READY LINE TO GO LOW AND THEN
OUTPUTS REG C TO DIDATA PORT
WAITOUT EQU $+SHIFT
IN DISTAT ;GET STATUS
ANI DREADY ;GET READY BIT
JNZ WAITOUT ;WAIT FOR READY
MOV A,C ;GET DATA IN A
OUT DIDATA ;OUTPUT IT
RET ;DONE

TURN ROUTINE WATCHES IF LINE WHEN
TURNAROUND OCCURS, AND WAITS FOR READY
DELAY IS INSERTED

TURN EQU $+SHIFT
IN DISTAT ;GET STATUS BYTE
ANI DIFACT OR DREADY ;GET IF STATUS
JNZ TURN ;WAIT FOR IF LOW

MVI A,DELAY/4 ;WAIT FOR TURNAROUND TO SETTLE

DELAY2 EQU $+SHIFT
DCR A
JNZ DELAY2
RET

THE ADDRESS ROUTINE CONVERTS TRACK, SECTOR,
AND DRIVE INFO INTO THE DRIVE #, LOW AND
HIGH ORDER ADDRESSES FOR THE 7710

ADDRESS EQU $+SHIFT
LDA TRACK ;GET TRACK
MOV C,A ;SAVE IT

PTXOO EQU $+SHIFT ;SETUP FOR PATCH LOCATION
LDA BTABLE ;GET MAX NUMBER OF SECTORS/TRACK

THE FOLLOWING CALCULATES THE RELATIVE OFFSET
AS TRACK*MAXSECTOR+SEC

MVI B,0 ;CLEAR HIGH BYTE
MVI L,0 ;CLEAR PRODUCT LOW BYTE
MOV H,A ;MOVE MULTIPLIER TO HIGH PRODUCT AREA
MVI A,B ;LOOP COUNTER

LOOP EQU $+SHIFT
DAD H ;SHIFT MULTIPLIER AND PRODUCT LEFT
JNC SKIP ;TEST A MULTIPLIER BIT
DAD B ;ADD MULTIPLICAND

SKIP EQU $+SHIFT
DCR A
JNZ LOOP ; 8 TIMES THROUGH THE LOOP
LDA SECTOR ; GET CURRENT SECTOR
MOV E,A ; PUT IN D
MVI D,0
DAD D ; ADD SECTOR TO OFFSET
XCHG ; PUT RELATIVE ADDRESS BACK IN DE
LHLD OFFSET ; GET OFFSET TO DISC #
DAD D ; GET PARTIAL OFFSET
XCHG ; PUT BACK IN DE
LHLD OFFSET ; TOTAL OFFSET IS 2X TABLE VALUE
DAD D ; GET TOTAL ADDRESS IN HL, WITH CARRY-HIGH BIT
MVI B,10H ; ASSUME HIGHEST ORDER BIT IS 1
RC ; WE WERE RIGHT
MVI B,0 ; HIGHER ORDER BIT IS A '0'
RET ; DONE

; ;
THIS ROUTINE PROCESSES THE HOME FUNCTION FOR THE HARD DISK
HHOME EQU $+SHIFT
MVI A,0 ; FOR HARD DISK JUST SET TRACK=0
STA TRACK ; FOR HARD DISK THATS ALL
RET ;

; ;
THIS ROUTINE INTERCEPTS THE TRACK SELECT ROUTINE
SETTRK EQU $+SHIFT
MOV A,C ; GET THE TRACK #
STA TRACK ; SAVE IT
JMP FSTTRK ; FINISH PROCESSING

; ;
THIS ROUTINE INTERCEPTS THE SECTOR SELECT ROUTINE
SECSET EQU $+SHIFT
MOV A,C ; GET THE SECTOR IN A
STA SECTOR ; STORE IT
JMP FSTSEC ; FINISH PROCESSING

; ;
THIS ROUTINE SETS THE LOCAL DMA ADDRESS
SETDMA EQU $+SHIFT
MOV H,B ; GET DMA IN (H,L)
MOV L,C
SHLD DMAADD ; STORE IT AWAY FOR LATER
JMP FSTDMA ; FINISH UP

; ;
SELECT EQU $+SHIFT
MOV A,C ; GET DRIVE # IN A
STA DRIVEN ; STORE IT FOR LATR
MVI B,0 ; (B,C)=DISC #
MOV L,C
MOV H,B ; HL=DISK#
DAD H
DAD H
DAD H ; HL=5*DISK NUMBER
XCHG ; DE=5*DISK NUMBER
LXI H,SIZTAB ; POINTER TO DISK PARAMETER TABLE
DAD D       ;POINT TO SELECTED DISK PARAMETERS
MOV A,M    ;GET LOW ORDER OFFSET FOR LOGICAL SECTOR
STA OFFSET ;STORE IT
INX H      ;POINT TO NEXT ENTRY
MOV A,M    ;GET HIGH ORDER OFFSET FOR LOGICAL SECTOR
STA OFFSET+1;STORE IT
INX H      ;POINT TO NEXT ENTRIES
MOV A,M    ;GET DRIVE NUMBER (PHYSICAL)
STA PDRIVE ;STORE IT
INX H      ;NEXT ENTRY
MOV E,M    ;GET POINTER TO DISK PARAMETER TABLE
INX H
MOV D,M
XCHG
    ;HL--> POINTER TO DISK SIZE TABLE
WVI 8,7   ;LENGTH OF TABLE TO COPY
PTX0
EQU $+SHIFT ;SETUP PATCH LOCATION
LXI H,BTABLE ;POINT TO DRIVE TABLE IN BDOS
MOV A,C    ;GET DISK NUMBER
CPI FMAX   ;IS IT A FLOPPY?
JC FLOPPY  ;YES - GO PROCESS IT
                    ;
                    ;THIS IS CODE FOR HARD DISK ONLY
                    ;
LDA DFLG    ;GET PREVIOUS DRIVE TYPE
ORA A       ;WAS IT A FLOPPY
JNZ SELI    ;NO, SO DO NOT OVERLAY TABLE
PUSH H      ;SAVE BIOS TABLE ADDRESS
PUSH D      ;SAVE CP/H BDOS TABLE ADDRESS
XCHG
    ;GET BDOS TABLE ADD. IN (H,L)
LXI D,FSIZE ;POINT TO BUFFER FOR FLOPPY TABLE
CALL COPY   ;SAVE COPY OF FLOPPY TABLE
WVI 8,7     ;SET TABLE SIZE AGAIN
POP D       ;GET BACK POINTERS
POP H
SELI
EQU $+SHIFT
CALL COPY   ;COPY HARD DISC TABLE INTO CP/M
LXI H,SCSET ;NO INTERLACE FOR HARD DISK
PTXI
EQU $+SHIFT ;SETUP ADDRESS PATCH LOCATION
SHLD CILACE
WVI A,1     ;SET FLAG FOR HARD DISC
STA DFLG
LXI H,HTAB  ;LOCATION OF HARD DISC TABLE
PTX4
EQU $+SHIFT
LXI D,SHOME ;POINT TO SWITCH TABLE (DESTINATION OF COPY)
WVI H,9     ;LENGTH OF TABLE
COPY
EQU $+SHIFT
MOV A,N     ;GET BYTE
STAX D      ;MOVE IT
INX H
INX D
DCR B       ;COUNT DOWN #
JNZ COPY
RET
          ;FLOPPY EQU $+SHIFT
FILE: CLINK  ASM  PAGE 011

CALL COPY ; COPY FLOPPY TABLE BACK INTO CP/M
PTX3 EQU $+SHIFT
LXI H,ILACE ; GET INTERLACE ROUTINE BACK
PTX2 EQU $+SHIFT ; SETUP PATCH LOCATION
SHLD CILACE ; STORE IT
LXI H,FTAB ; POINT TO FLOPPY JUMP TABLE
CALL PTX4 ; COPY IT INTO BIOS
XRA A ; CLEAR FLAG FOR FLOPPY
STA DFLG
JMP FSELEC ; LET NORMAL FLOPPY BIOS ROUTINE PROCESS
; THE REST

PDRIVE EQU $+SHIFT ; STORAGE FOR PHYSICAL DRIVE NUMBER
DB 0
DMAADD EQU $+SHIFT ; STORAGE FOR DMA ADDRESS
DW 0
TRACK EQU $+SHIFT ; STORAGE FOR CURRENT TRACK
DB 0
OFFSET EQU $+SHIFT ; STORAGE FOR OFFSET/2
DW 0
SECTOR EQU $+SHIFT ; STORAGE FOR CURRENT SECTOR
DB 0
DRIVEN EQU $+SHIFT ; STORAGE FOR DRIVE SELECT
DB 0
DFLG EQU $+SHIFT ; FLAG FOR PREVIOUS DRIVE TYPE
DB 0

MAXDRV EQU 14 ; NUMBER OF CONFIGURED DRIVES

--- THIS TABLE GETS PATCHED ON STARTUP TO MATCH THE FLOPPY
; TABLE IN BDOS (BTABLE).

FSIZE EQU $+SHIFT ; SECTORS/TRACK FOR FLOPPY
DB 26 ; # OF DIRECTORY ENTRIES
DB 63 ; FLOPPY BLOCK SIZE PARAMETERS
DB 3,7 ; MAX # OF BLOCKS ON DISK
DB 0F2H ; DIRECTORY ALLOCATION
DB 0CON
DB 2 ; NUMBER OF TRACKS FOR BOOT AND OPERATING SYS

H512 EQU $+SHIFT ; 512 K BYTE DISK (OFFSET FACTOR 2048)
DB 255 ; SECTORS/TRACK ON HARD DISK (ARBITRARY)
DB 255 ; # OF DIRECTORY ENTRIES
DB 4,15 ; BLOCK SIZE PARAMETERS (2K BLOCKS)
DB 255 ; # OF BLOCKS ON DISK
DB 0FH ; DIRECTORY ALLOCATION
DB 0 ; NUMBER OF BOOT TRACKS

H3824 EQU $+SHIFT ; 3.8 MBYTE DISK (OFFSET FACTOR 16384)
DB 255 ; SECTORS/TRACK
DB 255 ; # OF DIRECTORY ENTRIES
FILE: CLINK  ASM  PAGE 012

DB  7,127 ;BLOCK SIZE PARAMATERS (16K BLOCKS)
DB  238 ;# OF BLOCKS ON DISK
DB  80H ;DIRECTORY ALLOCATION
DB  0 ;NUMBER OF BOOT TRACKS

; SIZTAB EQU $+SHIFT
DW  0 ;OFFSET FOR DRIVE A
DB  0 ;PHYSICAL DRIVE #
DW  FSIZE ;DRIVE A=FLOPPY

; DW  0 ;OFFSET/2 FOR DRIVE B
DB  0 ;PHYSICAL DRIVE #
DW  FSIZE ;DRIVE B=FLOPPY

; DW  0 ;OFFSET/2 FOR DRIVE C
DB  1 ;PHYSICAL DRIVE FOR C
DW  H512 ;512 KBYTE DRIVE TABLE

; DW  2048 ;OFFSET/2 FOR DRIVE D
DB  1 ;PHYSICAL DRIVE FOR D
DW  H512 ;512 KBYTE DRIVE TABLE

; DW  4096 ;OFFSET/2 FOR DRIVE E
DB  1 ;PHYSICAL DRIVE FOR E
DW  H512 ;512 KBYTE DRIVE TABLE

; DW  6144 ;OFFSET/2 FOR DRIVE F
DB  1 ;PHYSICAL DRIVE FOR F
DW  H512 ;512 KBYTE DRIVE TABLE

; DW  8192 ;OFFSET/2 FOR DRIVE G
DB  1 ;PHYSICAL DRIVE FOR G
DW  H512 ;512 KBYTE DRIVE TABLE

; DW  10240 ;OFFSET/2 FOR DRIVE H
DB  1 ;PHYSICAL DRIVE FOR H
DW  H512 ;512 KBYTE SIZE TABLE

; DW  12288 ;OFFSET/2 FOR DRIVE I
DB  1 ;PHYSICAL DRIVE FOR I
DW  H512 ;512 KBYTE SIZE TABLE

; DW  14336 ;OFFSET/2 FOR DRIVE J
DB  1 ;PHYSICAL DRIVE FOR J
DW  H512 ;512 KBYTE SIZE TABLE

; DW  16384 ;OFFSET/2 FOR DRIVE K
DB  1 ;PHYSICAL DRIVE FOR K
DW  H512 ;512 KBYTE SIZE TABLE

; DW  18432 ;OFFSET/2 FOR DRIVE L
DB  1 ;PHYSICAL DRIVE FOR L
DW  H512 ;512 KBYTE SIZE TABLE

; DW  20480 ;OFFSET/2 FOR DRIVE M
DB  1 ;PHYSICAL DRIVE FOR M
; 512 KBYTE SIZE TABLE
DB 1 ; PHYSICAL DRIVE FOR N
DW 22528 ; OFFSET/2 FOR DRIVE N
DW H3824 ; 3.8 MBYTE SIZE TABLE

ENDP EQU $   ; LENGTH OF CODE TO COPY
LENCP EQU ENDP-START
MSIZE EQU 22 ; PUT REAL CPM MEMORY SIZE HERE
SBIOS EQU 1F00H ; LOCATION OF BIOS IN SYSGEN IMAGE
; (LIFEBOATS CP/M 1.44 FOR NORTHSTAR)
; STD VALUE FOR 8 INCH SINGLE DENSITY
; CP/M 1.4 IS 1E80H

DELTA EQU 000H ; OFFSET FROM STD CP/M SIZE
BIAS EQU (MSIZE-16)*1024-DELTA

FLOPPY EQU 313AH+BIAS ; POINTER TO THE PARAMETER TABLE FOR
; FLOPPY DISK PARAMETERS

FJUMP EQU 3E18H+BIAS ; THIS IS THE LOCATION OF JUMP VECTORS FOR
; FLOPPY DISK (STARTING AT JMP FHOME)

OFFSET EQU SBIOS-3E00H-BIASH ; OFFSET FOR READING PATCH. HEX IN DDT

END
FILE: DIR  SUB  PAGE 001

DIR C:*.*, $1
DIR D:*.*, $1
DIR E:*.*, $1
DIR F:*.*, $1
DIR G:*.*, $1
DIR H:*.*, $1
DIR I:*.*, $1
DIR J:*.*, $1
DIR K:*.*, $1
DIR L:*.*, $1
DIR M:*.*, $1
DIR N:*.*, $1
FILE: PUTGET  ASM  PAGE 001

------- CORVUS PUT/GET PROGRAM FOR CP/M -------
VERSION 1.2
BY BRK

THIS PROGRAM PERFORMS THREE TASKS:

1. PUT: TRANSFER A BLOCK OF CODE FROM MEMORY TO DISC.
2. GET: TRANSFER A BLOCK OF CODE FROM DISC TO MEMORY.
3. FILL: FILL A CONTIGUOUS SECTION OF THE DISC WITH A
   SPECIFIED BYTE.

--- COMMENTS ON PROGRAM INPUTS:

1. THE DRIVE #, DISC ADDRESS (0-75743), AND # OF SECTORS
   ARE ALL IN DECIMAL. THE PROGRAM IS SETUP FOR 128 BYTE
   SECTORS. THE DISC ADDRESS IS A NUMBER FROM 0 TO 75743
   (FOR THE 10MBYTE DRIVE) WHICH IS USED TO NUMBER ALL OF
   THE 128 BYTE SECTORS.
2. THE STARTING RAM ADDRESS IS IN HEX.
3. A CONTROL-C INPUT IN RESPONSE TO THE PUT/GET/FILL QUERY
   WILL CAUSE A RETURN TO CP/M (WITHOUT RE-BOOTING).
4. A CONTROL-C INPUT IN RESPONSE TO OTHER QUERIES WILL CAUSE
   A BRANCH TO THE PUT/GET/FILL QUERY.
5. AN INVALID INPUT WILL EITHER BE IGNORED, CAUSE A REPEAT
   OF THE QUESTION, OR RESULT IN AN ERROR MESSAGE.
6. THE FILL COMMAND IS CAPABLE OF FILLING THE ENTIRE DISC
   WITH A SPECIFIED BYTE. HOWEVER, THIS WOULD TAKE NEARLY
   AN HOUR TO DO SO. IT IS MAINLY USEFUL FOR FILLING
   SMALLER SECTIONS OF THE DISC (SUCH AS FILLING THE CP/M
   DIRECTORY AREAS WITH 0E5H ).
7. AFTER EACH SECTOR IS READ OR WRITTEN, THE CONSOLE STATUS
   IS CHECKED. IF A CONTROL-C HAS BEEN ISSUED, THE DISC
   OPERATION WILL BE ABORTED. IF SOME OTHER CHARACTER HAS
   BEEN HIT, A MESSAGE WILL BE DISPLAYED INDICATING THAT
   A DISC OPERATION IS STILL IN PROGRESS (THIS IS USEFUL
   ON LONG PUT OR FILL OPERATIONS TO SHOW THAT
   SOMETHING IS REALLY HAPPENING).

NOTE: THIS PROGRAM IS AN UPDATED VERSION OF PUTGET VERSION 1.0.
MODIFICATIONS FROM THE OLDER VERSION INCLUDE:

1. ADDITION OF THE FILL COMMAND.
2. CHANGING THE READ/ WRITE COMMANDS TO THE NEW
   VARIABLE SECTOR SIZE COMMAND FORMAT INTRODUCED
   WITH "THE MIRROR".
3. DOWNWARDS COMPATIBILITY WITH THE ORIGINAL 128 BYTE/SEC
   CONTROLLER CODE BY READING THE CONTROLLER CODE VERSION #
   AND PATCHING THE READ/ WRITE COMMANDS APPROPRIATELY.
4. CHANGING THE MAXIMUM DISC SIZE TESTS TO REFLECT THE SIZES
   SUPPORTED BY "THE MIRROR".

--- CP/M EQUATES -----

BDOS  EQU  05     ; BDOS ENTRY POINT
FILE: PUTGET ASM PAGE 002

; CR EQU ODH ; CARRIAGE RETURN
LF EQU OAH ; LINE FEED

; ---- CORVUS DISC EQUATES ----

DATA EQU UD8H ; DATA I/O PORT
STAT EQU DATA+1 ; STATUS INPUT PORT
DRDY EQU 1 ; MASK FOR DRIVE READY BIT
DIFAC EQU 2 ; MASK FOR DRIVE ACTIVE BIT

; ---- DO NOT CHANGE RDCOM OR WRCOM WITHOUT ALSO CHANGING THE TEST
; AT THE END OF THE INIT ROUTINE. ----

RDCOM EQU 12H ; READ COMMAND (MIRROR COMPATIBLE)
WRCOM EQU 13H ; WRITE COMMAND (MIRROR COMPATIBLE)

VERCOM EQU 0 ; COMMAND TO READ VERSION # AND # DRIVES
MAXSI EQU 0EOH ; MAXSI-MAXS3: MAX # OF SECTORS ON DISC
MAXS2 EQU 27H ; NOW SET AT 75743+1
MAXS3 EQU 1
SSIZE EQU 128 ; SECTOR SIZE (IN BYTES)
MAXDRV EQU 4 ; MAX # OF DRIVES

ORG 100H ; STANDARD CP/M TPA ORIGIN

START: LXI H,0
DAD SP ; GET STACK POINTER IN (H,L)
SHLD SBUF ; SAVE IT

; -- SETUP DIRECT CONSOLE I/O JUMPS ---
LHLD 1 ; GET ADDRESS OF WARM BOOT (BIOS+3)
LXI D,3
DAD D ; COMPUTE ADDRESS OF C
SHLD CONST+1 ; PATCH IN JUMP
DAD D
SHLD CONIN+1
DAD D
SHLD CONOUT+1
JMP SIGNON ; SIGN ON AND START PROGRAM

CONST: JMP 0 ; JUMP TO BIOS ROUTINES
CONIN: JMP 0
CONOUT: JMP 0

SIGNON: LXI SP,STACK ; SETUP LOCAL STACK
LXI D,SHMSG ; POINT TO MESSAGE
CALL PTHMSG ; PRINT SIGN ON MESSAGE
PGQ: LXI D,PQMSG
CALL PTHMSG ; ASK IF PUT OR GET
PI: CALL CIN ; GET CONSOLE CHAR.
CPI 'C'-40H ; IS IT A CONTROL-C ?
JNZ PGQ1 ; NO, SO CONTINUE
CEXIT: LXI D,CMMSG ; YES, SO ISSUE MESSAGE AND EXIT PROGRAM
CALL PTHMSG
LHLD SBUF ; GET OLD STACK POINTER
FILE: PUTGET  ASH  PAGE 003

SPHL
RET            ; RE-ENTER CP/M

; PGQ1:
CPI 'G'        ; IS IT A GET COMMAND?
MVI B,RDCOMH  ; GET READ COMMAND
JZ  PGQ2
CPI 'P'        ; IS IT A PUT COMMAND?
MVI B,WRCOMH  ; GET WRITE COMMAND
JZ  PGQ2
CPI 'F'        ; IS IT A FILL COMMAND?
JNZ PI         ; IF INVALID, GET ANOTHER CHAR.

; PGQ2:
STA COMD       ; SAVE COMMAND FOR REF.
MOV A,B       ; GET READ / WRITE DISC COMMAND
STA RWCOM     ; SAVE IT
CALL COUT     ; ECHO VALID COMMAND

; --- GET DRIVE # ---

; GTDRV:
LXI D,DMSG
CALL PTMSG    ; ASK FOR DRIVE #

; GT1:
CALL CIN
CPI 'C'~40H    ; IS IT A CONTROL-C
JZ  PGQ       ; YES, SO RESTART
SUI '0'       ; REMOVE ASCII BIAS
JC  GT1       ; IF INVALID, GET ANOTHER CHAR
JZ  GT1
CPI MAXDRV+1   ; TEST IF DRIVE # TO LARGE
JNC GT1
STA DRIVE     ; SAVE DRIVE #
CALL COUT     ; ECHO CHARACTER

; LDA COMD      ; GET PUT, GET, FILL COMMAND
CPI 'F'        ; WAS IT A FILL COMMAND?
JNZ GTAD      ; NO, SO ASSUME PUT OR GET

; --- GET FILL BYTE ---

; GTFIL:
LXI D,FMSG     ; ASK FOR FILL BYTE
CALL PTMSG
CALL INHEX
JC  GTFIL
XRA A
CMP H         ; IS UPPER BYTE 0?
JNZ GTFIL     ; NO, TRY AGAIN
MOV A,L       ; GET BYTE
STA FILLB     ; SAVE IT
JMP GTDAD

; --- GET DMA START ADDRESS ---

; GTAD:
LXI D,AMSG    ; ASK FOR MEMORY ADDRESS
CALL PTMSG
CALL INHEX
JC  GTAD      ; IF ERROR, ASK AGAIN
SHLD RADD     ; SAVE ADDRESS

; --- GET STARTING DISC ADDRESS (DECIMAL) ---
FILE: PUTGET ASM PAGE 004

; GTDAD: LXI D,DDMSG
    CALL PTMSG ; ASK FOR DISC ADDRESS
    CALL INDEC
    JC GTDAD ; IF INVALID, ASK AGAIN
    LXI H,CONV ; POINT TO CONVERSION BUFFER
    LXI D,DADD ; POINT TO BUFFER FOR DISC ADDRESS
    CALL COPY3 ; COPY TO BUFFER

; --- GET # OF SECTORS ----

; GTNS: LXI D,BMSG ; ASK FOR # OF SECTORS
    CALL PTMSG
    CALL INDEC
    JC GTNS ; IF INVALID, ASK AGAIN
    LXI H,CONV ; POINT TO CONVERSION BUFFER
    LXI D,NBLKS ; POINT TO BUFFER FOR # OF SECTORS
    CALL COPY3 ; COPY TO BUFFER
    LXI H,NBLKS+2 ; POINT TO THIRD BYTE OF # SECTORS
    XRA A ; CLEAR A
    ORA H
    DCX H
    ORA H
    DCX H
    ORA H
    JZ GTNS ; IF # SECTORS = 0
    LXI H,NBLKS
    LXI D,DADD
    CALL ADDM ; ADD # SEC AND DISC ADDRESS
    LXI D,MAXSC
    LXI H,ABUF ; SUBTRACT RESULT FROM MAX DISC ADD.+1
    CALL SUBM
    JC ROLD ; IF, TOO BIG

    LDA COMD ; GET PUT, GET, FILL COMMAND
    CPI 'F' ; IS IT A FILL COMMAND?
    JZ OK ; YES, SO TESTS ARE DONE

    LDA NBLKS+2 ; GET UPPER BYTE OF SECTOR COUNT
    ORA A
    JNZ ROLL ; IF FAR TOO BIG, ISSUE ERROR MESSAGE
    LXI B,-1 ; SETUP TO TEST IF MEMORY ROLLOVER
    LHLD RADD ; GET RAM ADD
    LXI D,SIZE

GTN1: DAD D ; LOOP TO FIND # SECTORS THAT COULD FIT
    INX B ; INC SECTOR COUNTER
    JNC GTN1
    MOV A,H
    ORA L
    JNZ GTN2 ; IF NOT EXACTLY ZERO
    INX B ; IF EXTRA SECTOR JUST FITS

GTN2: LHLD NBLKS ; COMPUTE #FIT-#SEC
    MOV A,C
    SUB L
    MOV A,B
    SBB H
    JNC OK ; OK SO CONTINUE
FILE: PUTGET    ASM    PAGE 005

ROLL:
   LXI   D,RLMSG ; ERROR IF ROLL OVER TOP OF MEMORY
   CALL  PTMSG
   JMP   GTNS
ROLD:
   LXI   D,RLDMSG ; IF POSSIBLE ROLL OVER DISC TOP
   CALL  PTMSG
   JMP   GTNS
   
   ; -- INPUTS ARE NOW ASSUMED TO BE VALID, SO SETUP TO DO OPERATION
   ; -- MERGE UPPER DISC ADDRESS NIBBLE WITH DRIVE #
OK:
   LDA   DADD+2
   ANI   OFH
   RLC
   RLC
   RLC
   RLC
   LXI   H,DRIVE
   ORA   M
   MOV   M,A
   CALL  INIT ; INITIALIZE CONTROLLER
   
   ; -- DO BLOCK OPERATION ---
   ;
   ; BLOCK:
   ; LHL   RADD ; GET RAM ADDRESS
   ; CALL  RWSEC ; READ OR WRITE ONE SECTOR
   ; SHLD  RADD
   ;
   ; CALL  CONST ; WAS A KEY HIT?
   ; ORA   A
   ; JNZ   BLK3 ; YES, SO ISSUE MESSAGE OR ABORT
   ;
   ; BLK1:
   ; LHL   NBLKS
   ; DCX   H
   ; SHLD  NBLKS
   ; MOV   A,H
   ; ORA   L
   ; JNZ   BLK2 ; NOT DONE YET, SO CONTINUE
   ; LXI   H, NBLKS+2 ; POINT TO UPPER BYTE OF SECTOR COUNT
   ; ORA   M ; TEST IF ZERO
   ; JZ    PQQ ; DONE, SO RETURN TO FIRST QUESTION
   ; DCR   M ; DECREMENT COUNT AND CONTINUE
   ;
   ; BLK2:
   ; LHL   DADD ; GET DISC ADDRESS
   ; LXI   D,1
   ; DAD   D
   ; SHLD  DADD ; UPDATE IT
   ; JNC   BLOCK ; DO ANOTHER SECTOR
   ; LDA   DRIVE
   ; ADI   10H ; IF CARRY, INCREMENT ADDRESS NIBBLE
   ; STA   DRIVE
   ; JMP   BLOCK
   ;
   ; BLK3:
   ; CALL  CONTIN ; GET INPUT CHAR.
   ; ANI   5FH ; MASK TO UPPER CASE
   ; CPI   'C'-'40H ; IS IT A CONTROL-C?
   ; LXI   D,MSG1 ; POINT TO MESSAGE
FILE: PUTGET

JNZ BLKA
LXI D, MSG2 ; POINT TO MESSAGE

BLKA:
PUSH PSW ; SAVE FLAGS
 CALL PTMSG ; PRINT MESSAGE
 POP PSW ; RESTORE FLAGS
 JNZ BLKI ; RETURN IF NOT CONTROL-C
 JMP PGQ ; RESTART MENU SELECTION

RWSEC:
LDA RWCOM ; GET COMMAND
 CALL WAITO ; WAIT AND SEND IT
 LDA DRIVE ; GET DRIVE #
 CALL WAITO
 LDA DADD ; GET LOW BYTE OF DISC ADDRESS
 CALL WAITO
 LDA DADD+1 ; GET UPPER BYTE OF DISC ADDRESS
 CALL WAITO
 LDA COMD ; GET COMMAND
 CPI 'F' ; WAS IT A FILL COMMAND?
 JZ FILL ; YES, SO FILL A SECTOR
 CPI 'P' ; WAS IT A PUT COMMAND?
 JZ WRIT ; YES, SO WRITE A SECTOR
 CALL WERR ; NO, SO ASSUME READ AND GET ERROR CODE

RSEC: MVI B, SSIZE

RLP:
 IN STAT ; READ STATUS PORT
ANI DRDY ; LOOK AT READY LINE
JNZ RLP ; LOOP UNTIL READY
 IN DATA ; READ BYTE FROM DISC
MOV M, A ; SAVE IT IN MEMORY
INX H
DCR B ; DECREMENT BYTE COUNT
JNZ RLP ; LOOP UNTIL DONE
 RET

FILL: MVI B, SSIZE
LDA FILLB ; GET FILL BYTE
MOV C, A ; INTO (C)

FLP:
 IN STAT ; READ STATUS PORT
ANI DRDY
JNZ FLP
MOV A, C ; GET FILL BYTE
OUT DATA ; WRITE IT TO DISC
DCR B
JNZ FLP ; LOOP UNTIL DONE

WRIT: MVI B, SSIZE

WLP:
 IN STAT ; READ STATUS PORT
ANI DRDY
JNZ WLP
MOV A, M ; GET BYTE FROM MEMORY
OUT DATA ; WRITE IT TO DISC
INX H
DCR B
JNZ WLP ; LOOP UNTIL DONE

WERR:
 CALL TURN ; TURN AROUND BUS
 CALL WAITI ; WAIT FOR ERROR BYTE
 MOV B, A ; SAVE BYTE
ANI $80H ; LOOK FOR FATAL ERRORS
RZ ; OK, SO RETURN
PUSH B ; SAVE ERROR
LXI D, $ERMSG ; ERROR, SO ISSUE MESSAGE
CALL PTMSG
POP PSW ; GET ERROR BYTE BACK IN ACC
CALL HEXOT ; OUTPUT IN HEX
LXI D, $ERMSG1
CALL PTMSG
JMP $SIGNON ; RESTART PROGRAM

TURN: IN STAT ; LOOK AT BUSS ACTIVE BIT
ANI $DIFAC ; TURN
MVI $B, 6 ; GOOD AT 4MHZ ALSO
DELAY: DCR B
JNZ DELAY
RET

WAITI: IN STAT ; READ STATUS PORT
ANI $DRDY ; LOOK AT READY LINE
JNZ WAITI ; LOOP UNTIL READY
IN DATA - ; READ BYTE FROM DISC
RET

WAITO: PUSH PSW ; SAVE COMMAND
IN STAT ; READ STATUS PORT
ANI $DRDY ; LOOK AT READY LINE
JNZ WAITO+1 ; LOOP UNTIL READY
POP PSW
OUT DATA ; WRITE BYTE TO DISC
RET

--- INITIALIZE CONTROLLER ---

INIT: MVI $A, $OFFH ; GET AN INVALID COMMAND
OUT DATA ; SEND IT TO CONTROLLER
MVI $B, $150 ; SET FOR LONG DELAY
CALL DELAY
IN STAT
ANI $DIFAC ; LOOK AT DRIVE ACTIVE BIT
JNZ INIT ; LOOP UNTIL NOT ACTIVE
CALL WAITI ; GET ERROR CODE
CPI $8FH ; CHECK RETURN CODE
JNZ INIT ; IF NOT RIGHT, TRY AGAIN

--- DETERMINE IF OLDER CONTROLLER CODE ---

MVI $A, $VERCOM ; GET VERSION # COMMAND
CALL WAITO ; SEND IT TO CONTROLLER
CALL TURN ; WAIT FOR BUSS TURN AROUND
CALL WAITI ; READ VERSION # AND # DRIVES
ANI $0FH ; MASK OFF # DRIVES
RNZ ; RETURN IF NEW CODE
LDA $RWCOM ; GET N/W COMMAND SELECTED
SUI $10H ; REMOVE SECTOR SIZE SELECT
STA $RWCOM ; SAVE IT BACK IN BUFFER
RET

; --- COPY ROUTINE ---

COPY3: MVI C, 3
COPY: MOV A, M
STAX D
INX H
INX D
DCR C
JNZ COPY
RET

; --- MULTI BYTE ADDITION ---
; (H, L) AND (D, E) POINT TO ADDENDS
; RESULT IS PUT IN CONVERSION BUFFER: ABUF

ADDM: PUSH H
PUSH D
PUSH B
LXI B, ABUF ; DESTINATION ADDRESS
PUSH B
MVI C, 3 ; ARITHMETIC PRECISION
XRA A ; CLEAR FLAGS
AD1: LDAX D
ADC M
XTXL
MOV M, A ; SAVE RESULT IN BUFFER
INX H
XTXL
INX H
INX D
DCR C
JNZ AD1 ; LOOP UNTIL DONE
POP B
POP B
POP D
POP H
RET

; --- MULTI BYTE SUBTRACTION ---
; (D, E) POINTS TO THE MINUEND
; (H, L) POINTS TO THE SUBTRAHEND
; [D, E] − [H, L]
; RESULT IS PUT IN CONVERSION BUFFER: ABUF

SUBM: PUSH H
PUSH D
PUSH B
LXI B, ABUF ; DESTINATION ADDRESS
PUSH B
MVI C, 3 ; ARITHMETIC PRECISION
XRA A ; CLEAR FLAGS
SD1: LDAX D
SBB M
XTXL
MOV M,A ; SAVE RESULT IN BUFFER
INX H
XTHL
INX H
INX D
DCR C
JNZ S01 ; LOOP UNTIL DONE
POP B
POP B
POP D
POP H
RET

; CIN:
PUSH H ; BUFFERED CONSOLE INPUT
PUSH D
PUSH B
CALL CONIN
POP B
POP D
POP H
MOV C, A ; SAVE FOR ECHO
CPI $40H ; IS IT LOWER CASE?
RC ; NO, SO RETURN
ANI $5FH ; YES, SO CONVERT TO UPPER CASE
RET

; COUT:
PUSH PSW ; SAVE ACC
PUSH H ; BUFFERED CONSOLE OUTPUT
PUSH D
PUSH B
CALL CONOUT
POP B
POP D
POP H
POP PSW
RET

; --- MESSAGE PRINT ROUTINE---

PTMSC: MVI C, 9 ; CP/M WRITE LIST COMMAND
CALL BDOS
RET

; --- OUTPUT BYTE IN ACC IN HEX ---

HEXOT: PUSH PSW ; SAVE BYTE
RRC
RRC
RRC
RRC
CALL HEXB ; OUTPUT UPPER NIBBLE IN HEX
POP PSW ; GET BYTE BACK
HEXB: ANI $0FH ; MASK OFF UPPER NIBBLE
ADI '0'; ADD ASCII BIAS
CPI '9'+1 ; TEST IF NUMERIC
JC PRT ; YES, SO DO IT
ADI 7 ; NO, SO ADD BIAS FOR A-F
FILE: PUTGET  ASH  PAGE 010

PRT: MOV  C,A ; SETUP FOR OUTPUT
     JMP  COUT ; OUTPUT HEX Nibble

; --- HEX INPUT ROUTINE ---

INHEX: LXI  H,0 ; CLEAR CONVERSION REGISTER
     CALL  CIN ; GET CHAR.
     CPI  'C'-40H
     JZ  RT1
     CPI  ' ' ; IS IT A SPACE
     JZ  H1 ; IGNORE IT
     CPI  CR ; IS IT A CR
     JNZ  HEX2
     GRA  A ; CLEAR FLAGS
     RET

HEX2: CALL  COUT ; ECHO CHARACTER
     SUI  '0' ; REMOVE ASCII BIAS
     RC
     CPI  'G'- '0'
     CMG
     RC
     CPI  10
     JC  HEX1
     SUI  7 ; ADJUST FOR A-F CHARACTERS
     CPI  10
     RC

HEX1: DAD  H ; SHIFT 16 BIT REGISTER OVER 4 PLACES
     DAD  H
     DAD  H
     ADD  H
     ADD  L ; ADD IN NEW NIBBLE
     MOV  L,A
     JMP  H1

RT1: POP  PSW ; CLEAR RETURN ADDRESS FROM STACK
     JMP  PGQ ; RETURN TO INITIAL QUERY

; --- 3 BYTE DECIMAL INPUT ROUTINE ---
; THE BINARY RESULT IS SAVED IN THE CONVERSION BUFFER: CONV

INDEC: LXI  H,CONV
     CALL  ZERO3 ; CLEAR BUFFER

IN1: CALL  CIN ; GET CHAR.
     CPI  'C'-40H
     JZ  RT1
     CPI  ' ';
     JZ  IN1 ; IGNORE SPACES
     CPI  CR ; IS IT A CR?
     JNZ  DEG2
     LXI  D,CONV
     LXI  H,MAXSC
     CALL  SUBH ; TEST IF # IS TOO BIG
     CMC
     RNC

BIG: LXI  D,BCMSG
     CALL  PTMSG ; ISSUE ERROR MESSAGE
     STC
     RET
DEC2: CALL COUT ; ECHO CHARACTER
SUI '0' ; REMOVE ASCII BIAS
RC
CPI 10
RC
DEC1: STA CONVX ; SAVE CHAR
LXI H,CONV
LXI D,CONV
CALL ADDM ; DOUBLE BUFFER VALUE
LXI H,ARUF
LXI D,ARUF
PUSH D
CALL ADDM ; DOUBLE IT AGAIN
LXI D,CONV
CALL ADDM ; NOW 5X STARTING VALUE
POP D
CALL ADDM ; NOW 10X STARTING VALUE
LXI D,CONV
CALL ADDM ; ADD IN NEW UNITS DIGIT VALUE
JC BIG ; IF CARRY OUT OF THIRD BYTE
LXI D,CONV
CALL COPY3 ; COPY TOTAL BACK TO CONV
JMP IN1 ; LOOP FOR MORE

; ZERO3: MVI C,3
ZERO: MVI M,0
INX H
DCR C
JNZ ZERO
RET

; ----- MESSAGES -----

SMSG: DB CR,LF,' CORVUS PUT/GET ROUTINE --',CR,LF,CR,LF
      DB ' ( VERSION 1.2 )',CR,LF,'$
PGMSG: DB CR,LF,' PUT, GET, OR FILL (P/G/F) ? $
DMSG: DB CR,LF,' DRIVE # (1-4) ? $
AMSG: DB CR,LF,' STARTING HEX RAM ADDRESS ? $
FMSG: DB CR,LF,' HEX BYTE TO FILL DISC WITH ? $
DDMSG: DB CR,LF,' STARTING DISC ADDRESS ? $
BMSG: DB CR,LF,' NUMBER OF SECTORS ? $
MSG1: DB CR,LF,CR,LF,' DISC OPERATION IN PROGRESS ',CR,LF,'$
MSG2: DB CR,LF,CR,LF,' -- DISC OPERATION ABORTED --',CR,LF,CR,LF,'$
BOMSG: DB CR,LF,CR,LF,07,' -- NUMBER IS TOO BIG -- ',CR,LF,'$
RLMSG: DB CR,LF,CR,LF,07,' -- THIS WOULD ROLL OVER THE TOP OF MEMORY --' DB CR,LF,'$
FILE: PUTGET ASM PAGE 012

; RDMSG: DB CR,LF,CR,LF,07,,' -- THIS WOULD EXCEED DISC SIZE -- ',CR,LF,'$'
; ERMSG: DB CR,LF,CR,LF,07,,' ** DISC R/W ERROR # $'
; ERMSG1: DB 'H **',CR,LF,'$'
; CMSG: DB 'C',CR,LF,'$'
; --- BUFFERS AND DATA ----
; MAXSC: DB MAXS1 ; MAXIMUM DISC ADDRESS
    DB MAXS2
    DB MAXS3
; CONVX: DB 0 ; BUFFER FOR INDEC ROUTINE
    DB 0
    DB 0
; SBUF: DS 2 ; OLD STACK POINTER
RWCMD: DS 1 ; R/W COMMAND
COND: DS 1 ; FUNCTION COMMAND (G, P, OR F)
DRIVE: DS 1 ; DRIVE # AND UPPER DISC ADDRESS NIBBLE
RADD: DS 2 ; RAM ADDRESS FOR DMA
DADD: DS 3 ; DISC ADDRESS
NBLKS: DS 3 ; # DISC SECTORS TO R/W
CONV: DS 3 ; CONVERSION BUFFER FOR INDEC
ABUF: DS 3 ; BUFFER FOR ADDM AND SUBM
FILLB: DS 1 ; FILL BYTE
    DS 80 ; STACK SPACE
STACK: NOP
; END
FILE: CLOADR ASM PAGE 001

; LOADER FOR CP/M ON CORVUS DISC
; VERSION 1.2
;
; THIS PROGRAM LOADS THE 1 SECTOR CP/M BOOT LOADER FROM DISC
; AND RUNS IT TO BOOT IN CP/M. IN THIS WAY, THIS LOADER IS
; INDEPENDENT OF THE SIZE OF CP/M.
;
; THIS PROGRAM MAY BE PUT IN ROM OR LOADED FROM FLOPPY OR CASSETTE
; (EVEN PAPER TAPE). THE EQUATES ARE NOW 511 UP SO THAT IT
; MAY BE LOADED UNDER CP/M.
;
; ----- CORVUS EQUATES ----
;
DATA EQU ODEH ; DATA I/O PORT
STAT EQU DATA+1 ; STATUS INPUT PORT
DROI EQU 1 ; MASK FOR DRIVE READY BIT
DIFAC EQU 2 ; MASK FOR DRIVE ACTIVE BIT
RDCCOM EQU 12H ; READ COMMAND (VERS. 1 CODE)
SSIZE EQU 128 ; SECTOR SIZE (IN BYTES)

BDRI EQU 1 ; DRIVE # TO BOOT FROM
BSEC EQU 12 ; DISC ADDRESS TO BOOT FROM (RESERVE A FEW SEC.)

CBOOT EQU 0 ; ORIGIN OF BOOT PROGRAM (THAT WHICH IS LOADED)

ORG 100H ; SET SO IT CAN BE LOADED FROM FLOPPY CP/M

START: LXI SP,OFFH ; PUT STACK IN A SAFE PLACE

; THE INIT ROUTINE INSURES THAT THE CONTROLLER STATE
; IS PROPERLY SETUP.

INIT: MVI A,OFFH ; GET INVALID COMMAND
OUT DATA ; SEND IT TO CONTROLLER
MVI B,150 ; SET FOR LONG DELAY
CALL DELAY
IN STAT ; READ STATUS
ANI DIFAC ; LOOK AT BUS ACTIVE BIT
JNZ INIT ; LOOP UNTIL OK
CALL WAITI ; READ POSSIBLE ERROR CODE
CPI 8FH ; TEST IT
JNZ INIT ; IF NOT CORRECT, DO IT AGAIN

READ: LXI H,CBOOT ; GET BOOT ADDRESS
LXI D,BSEC ; GET BOOT SECTOR ADDRESS
MVI A,RDCCOM ; GET READ COMMAND
CALL WAITO ; SEND IT TO CONTROLLER
MVI A,BDRI ; GET DRIVE #
CALL WAITO
MOV A,E ; GET LOW BYTE OF DISC ADDRESS
CALL WAITO
MOV A,D ; GET UPPER BYTE OF DISC ADDRESS
CALL WAITO
CALL TURN ; WAIT FOR BUS TO TURN AROUND
CALL WAITI ; READ ERROR CODE
ANI 80H ; LOOK AT FATAL BIT
JNZ START ; IF ERROR, TRY AGAIN
FILE: CLOADR  ASM  PAGE 002

MVI  B, SSIZE  ; GET SECTOR SIZE
PUSH  H  ; SAVE LOAD ADDRESS ON STACK
RLP:
CALL  WAITI  ; READ DATA BYTE FROM DISC
MOV  M, A  ; SAVE IT IN MEMORY
INX  H
DCR  B
JNZ  RLP  ; LOOP UNTIL DONE
RET  ; JUMP INTO CODE

; WAITO:
PUSH  PSW
CALL  DSTAT  ; WAIT FOR READY
POP  PSW
OUT  DATA  ; OUTPUT COMMAND
RET

; WAITI:
CALL  DSTAT
IN  DATA
RET

; DSTAT:
IN  STAT  ; READ STATUS WORD
ANI  DRY  ; LOOK AT READY BIT
JNZ  DSTAT  ; LOOP UNTIL READY
RET

; TURN:
IN  STAT  ; READ STATUS WORD
ANI  DIFAC  ; LOOK AT DRIVE ACTIVE BIT
JNZ  TURN  ; LOOP UNTIL DONE
MVI  B, 6  ; SET DELAY (GOOD AT 4MHz CLOCK)
DELAY:
DCR  B
JNZ  DELAY
RET

; END
FILE: CBOOT  ASH  PAGE 001

; BOOT ROUTINE FOR CP/M ON CORVUS DISC
; VERSION 1.21
;
; THIS PROGRAM WILL LOAD IN CP/M FROM THE CORVUS DISC.
; IT IS FIRST LOADED IN WITH THE "CLOADR" PROGRAM.
;
; THIS PROGRAM IS 1 SECTOR LONG AND MUST BE STORED ON DISC.
; IT MUST BE CHANGED WHENEVER THE CP/M SIZE IS CHANGED.
;
; ----- CORVUS EQUATES ----- 

DATA  EQU  0DEH ; DATA I/O PORT
STAT  EQU  DATA+1 ; STATUS INPUT PORT
DRDY  EQU  1 ; MASK FOR DRIVE READY BIT
DIFAC  EQU  2 ; MASK FOR DRIVE ACTIVE BIT
RDCOM  EQU  12H ; READ COMMAND (VERS. 1 CCODE)
SSIZE  EQU  128 ; SECTOR SIZE (IN BYTES)
;
BDRIVE  EQU  1 ; DRIVE # TO BOOT FROM
BSEC  EQU  12 ; DISC ADDRESS TO BOOT FROM (RESERVE A FEW SEC.)
CSEC  EQU  BSEC+1 ; STARTING SECTOR FOR CP/M ON DISC
;
; ----- CP/M EQUATES ----- 

MSIZE  EQU  20 ; CP/M MEMORY SIZE IN KB
DELT A  EQU  0000H ; OFFSET FROM STD CP/M SIZE
BIAS  EQU  (MSIZE-20)*1024-DELT A
CCP  EQU  3400H+BIAS ; CP/M LOAD ADDRESS (CP/M 2.0)
BIOS  EQU  CCP+1600H ; BASE OF BIOS (CP/M 2.0)
BOOT  EQU  BIOS  ; ENTRY POINT AFTER BOOT
NSEC  EQU  59 ; NUMBER OF SECTORS TO LOAD (COULD BE SMALLER)
;
CBOOT  EQU  0 ; ORIGIN OF BOOT PROGRAM
;
ORG  CBOOT
;
START:  LXI  SP,OFFH ; PUT STACK IN A SAFE PLACE
;
LDCHF:  MVI  C,NSEC  ; GET # SECTORS TO LOAD
         LXI  H,CCP  ; GET LOAD RAM ADDRESS
         LXI  D,CSEC  ; GET STARTING DISC ADDRESS

LD1:  CALL  READ ; READ IN ONE SECTOR
       INX  D  ; INCREMENT SECTOR COUNT
       DCR  C  ; COUNT DOWN # SECTORS
       JNZ  LD1  ; LOOP UNTIL DONE
       JMP  BOOT  ; IF DONE, ENTER CP/M
;
READ:  MVI  A,RDCOM  ; GET READ COMMAND
       CALL  WAITO ; SEND IT TO CONTROLLER

MVI  A,BDRIVE  ; GET DRIVE #
       CALL  WAITO
MOV  A,E  ; GET LOW BYTE OF DISC ADDRESS
       CALL  WAITO
MOV  A,D  ; GET UPPER BY DE OF DISC ADDRESS
       CALL  WAITO
       CALL  TURN  ; WAIT FOR BUSS TO TURN AROUND
CALL WAITI ; READ ERROR CODE
ANI 80H ; LOOK AT FATAL BIT
RD1: JNZ RD1 ; IF ERROR, LOOP
MVI B, SSIZE ; GET SECTOR SIZE.
RLP: IN STAT ; READ STATUS
ANI DRDY ; LOOK AT READY BIT
JNZ RLP ; LOOP UNTIL READY
IN DATA ; GET BYTE FROM DISC
MOV M, A ; SAVE IT IN MEMORY
INX H
DCR B
JNZ RLP ; LOOP UNTIL DONE
RET

WAITO: PUSH PSW
IN STAT ; READ STATUS WORD
ANI DRDY ; LOOK AT READY BIT
JNZ WAITO+1
POP PSW
OUT DATA ; OUTPUT COMMAND
RET

WAITI: IN STAT ; WAIT UNTIL READY
ANI DRDY
JNZ WAITI
IN DATA ; READ BYTE FROM DISC
RET

TURN: IN STAT ; READ STATUS WORD
ANI DIFAC ; LOOK AT DRIVE ACTIVE BIT
JNZ TURN ; LOOP UNTIL DONE
MVI B, 6 ; SET DELAY (GOOD AT 4MHZ CLOCK)
DELAY: DCR B
JNZ DELAY
RET

END
FILE: BIOSC   ASM   PAGE 001

; CORVUS DISC DRIVERS FOR CP/M 2.0   (BIOS)
;
;   VERSION 1.21
;
; MSIZE   EQU   20   ; CP/M VERSION MEMORY SIZE IN KB
;
; DELTA   EQU   0000H   ; OFFSET FROM STD CP/M SIZE
; BIAS    EQU   (MSIZE-20)*1024-DELTA   ; OFFSET FROM 20K CP/M
; CCP     EQU   3400H+BIAS   ; BASE OF CP/M
;
; OFFSET   EQU   9800H-CCP   ; OFFSET USED WITH DDT IN
;                    ; SYSTEM CONFIGURATION
; BDOS    EQU   CCP+806H   ; BASE OF BDOS
; BIOS    EQU   CCP+1600H   ; BASE OF BIOS
; NSEC    EQU   (BIOS-CCP)/128   ; # SECTORS TO BOOT
; CDISC   EQU   04   ; BUFFER LOCATION FOR CURRENT DISC #
; IBYTE   EQU   03   ; LOCATION OF INTEL 10BYTE
;
; ----- CORVUS EQUATES -----
;
; DATA     EQU   0DEH   ; DISC I/O PORT #
; STAT     EQU   DATA+1   ; DISC STATUS PORT
; DRY      EQU   1   ; MASK FOR DRIVE READY BIT
; DIFAC    EQU   2   ; MASK FOR DRIVE ACTIVE BIT
; RDCOM    EQU   12H   ; READ COMMAND (VERS. 1 CCODE)
; WRCOM    EQU   13H   ; WRITE COMMAND (VERS. 1 CCODE)
; NPSUDO   EQU   2   ; NUMBER OF PSEUDO DRIVES ON SINGLE CORVUS DRIVE
; DMAX     EQU   4   ; TOTAL # OF DRIVES (INCLUDES TWO 8 INCH FLOPPIES)
; SSIZE    EQU   128   ; SECTOR SIZE (IN BYTES)
; BDROVE   EQU   1   ; CORVUS DRIVE # TO BOOT FROM
; CSEC     EQU   13   ; STARTING DISC ADDRESS FOR CP/M BOOT
;
;   ORG   BIOS
;
; CP/M INTERFACE JUMP TABLE
;
; JMP   BOOT
;
; WBOOTE:   JMP   WROOT
; JMP   CONST
; JMP   CONIN
; JMP   CONOUT
; JMP   LIST
; JMP   PUNCH
; JMP   READER
; JMP   DHOME
; JMP   SELDSK
; JMP   SETTRK
; JMP   SETSEC
; JMP   SETDMA
; JMP   DREAD
; JMP   DWRT
; JMP   LISTST   ; LIST DEVICE STATUS REQUEST
; JMP   SECTRAN   ; SECTOR TRANSLATION ROUTINE
FILE: BIOSC ASM PAGE 002

; ---- DISC PARAMETER BLOCKS ----
; THE EXAMPLE HERE DIVIDES ONE 9.7MBYTE CORVUS DISC INTO
; TWO LARGE PSEUDO DRIVES (OF EQUAL SIZE)
; AND ALSO PROVIDES FOR THE INTERFACE OF TWO STANDARD 8 INCH
; SINGLE DENSITY FLOPPY DISC DRIVES.
;
; NOTE: THE NUMBERS SHOWN IN DPBC (THE PARAMETER BLOCK)
; FOR THE PSEUDO DRIVE AND ITS ASSOCIATED ALLOCATION
; BUFFER SIZES ARE THE RESULT OF CHOOSING:
; 37860 SECTORS/PSEUDO DRIVE
; 60 SECTORS/TRACK
; 1 RESERVED TRACK FOR OPERATING SYSTEM
; 256 DIRECTORY ENTRIES
; 8*1024 BYTE BLOCKS
;
; DPBASE EQU $;
; DPBO: DW 0,0 ; CORVUS PSEUDO DRIVE 1
; DW 0,0
; DW DIRBUF,DPBC ; DIRECTORY BUFFER, PARAM. BLOCK
; DW CV0,ALV0 ; CHECK, ALLOC MAP
; DPE1: DW 0,0 ; CORVUS PSEUDO DRIVE 2
; DW 0,0
; DW DIRBUF,DPBC ; DIRECTORY BUFFER, PARAM. BLOCK
; DW CV1,ALV1 ; CHECK, ALLOC MAP
; DPE2: DW FTAB,0 ; FLOPPY TRANSLATION TABLE
; DW 0,0
; DW DIRBUF,DPBF ; DIRECTORY BUFFER, PARAM. BLOCK
; DW CV2,ALV2 ; CHECK, ALLOC MAP
; DPE3: DW FTAB,0 ; FLOPPY TRANSLATION TABLE
; DW 0,0
; DW DIRBUF,DPBF ; DIRECTORY BUFFER, PARAM. BLOCK
; DW CV3,ALV3 ; CHECK, ALLOC MAP
; DPBC: DW 60 ; SECTORS/TRACK ON CORVUS PSEUDO DRIVE
; DB 6 ; BLOCK SHIFT
; DB 63 ; BLOCK MASK
; DB 3 ; EXTENT MASK
; DW 589 ; DISK SIZE-1
; DW 255 ; DIRECTORY MAX
; DB 128 ; ALLOC0
; DB 0 ; ALLOC1
; DW 0 ; CHECK SIZE
; DW 1 ; OFFSET
; DPBF: DW 26 ; SECTORS/TRACK ON STD 8 INCH FLOPPY
; DB 3 ; BLOCK SHIFT FACTOR
; DB 7 ; BLOCK MASK
; DB 0 ; NULL MASK
; DW 242 ; DISK SIZE-1
; DW 63 ; DIRECTORY MAX
; DB 192 ; ALLOC 0
; DB 0 ; ALLOC 1
FILE: BIOSC  ASH  PAGE 003

   DW    16 ; CHECK SIZE
   DW    2 ; TRACK OFFSET
;
; ----- CORVUS DISC OFFSET TABLE -----
;
   OFSBAS EQU   S
PDRVO: DW CSEC-1 ; STARTING DISC ADDRESS FOR DRIVE 0
   DB    0 ; THIS IS THE UPPER BYTE OF THE 20 BIT DISC ADDRESS
   DB    1 ; ACTUAL PHYSICAL DRIVE # (1-4)
;
PDRV1: DW 37884 ; STARTING DISC ADDRESS FOR DRIVE 1
   DB    0;
   DB    1 ; ACTUAL PHYSICAL DRIVE # (1-4)
;
; ----- STANDARD 8 INCH FLOPPY INTERLACE TABLE -----
;
   FTAB: DB 1,7,13,19
   DB 25,5,11,17
   DB 23,3,9,15
   DB 21,2,8,14
   DB 20,26,6,12
   DB 18,24,4,10
   DB 16,22
;
; ----- AUXILIARY JUMP TABLE FOR DRIVE SWITCHING -----
;
   DHOME: JMP HOMEC ; SET TO HOME CORVUS DISC DRIVE
   DREAD: JMP READC ; SET TO READ FROM CORVUS DRIVE
   DWRIT: JMP WRITC ; SET TO WRITE TO CORVUS DRIVE
;
; ----- SECTOR TRANSLATION ROUTINE -----
;
   SECTRAN: MOV A,D ; TEST IF TABLE TRANSLATION IS REQUESTED
   ORA E
   JNZ STRI ; YES, SO DO IT
   MOV L,C ; NO, SO JUST TRANSFER TO (H,L)
   MOV H,B
   RET

   STRI: XCHG ; GET TABLE ADDRESS IN (H,L)
   DAD B ; INDEX INTO TABLE
   MOV L,M ; GET BYTE IN (H,L)
   MVI H,0
   RET
;
; ----- COLD BOOT STARTUP -----
;
   BOOT: LXI SP,80H ; SETUP TEMP. STACK
   LXI H,BMSG ; POINT TO BOOT UP MESSAGE
   CALL PTMSG ; PRINT IT OUT
   MVI A,0 ; GET CURRENT DISC #
   STA CDISC ; SAVE IN BUFFER
   MOV C,A
   CALL SELDSK ; SELECT IT ALSO (INITIALIZE BUFFERS)

   GOCPM: MVI A,0C3H ; GET JUMP INSTRUCTION
   STA 0 ; SETUP FOR WARM BOOT
   LXI H,WBOOTE ; WARM BOOT ENTRY
   SHLD 1 ; SET ADDRESS
FILE: BIOSC  ASM  PAGE 004

STA 5 ; SETUP BDOS ENTRY JUMP
LXI H,BDOS
SHLD 6
LXI B,80H ; DEFAULT DMA ADDRESS
CALL SETDMA
LDA CDISC ; GET CURRENT DRIVE #
MOV C,A ; SAVE FOR CCP FUNCTION
JMP CCP ; ENTER CP/M

; ---- WARM BOOT STARTUP ROUTINE ----

WBOOT: LXI SP,80H ; SET STACK
WBO1: MV1 A,OFFH ; GET INVALID COMMAND
OUT DATA ; SEND IT TO CONTROLLER
MV1 B,150 ; SET FOR LONG DELAY
CALL DELAY
IN STAT ; GET STATUS BYTE
ANI DIFAC ; LOOK AT DRIVE ACTIVE BIT
JNZ WBO1 ; LOOP UNTIL NOT ACTIVE
CALL WAITI ; WAIT FOR ERROR CODE
CP1 $FH ; CHECK RETURN CODE
JNZ WBO1 ; IF NOT RIGHT, TRY AGAIN

MV1 C,NSEC ; GET # SECTORS TO BOOT
LXI H,CCF ; GET RAM START ADDRESS OF LOAD
LXI D,CSEC ; GET DISC ADDRESS FOR COPY OF CP/M
BT1: MV1 A,RDCOM ; GET READ COMMAND
CALL WAITO ; SEND IT TO CONTROLLER
MV1 A,BDRIVE ; GET BOOTUP DRIVE # (CORVUS PHYSICAL DRIVE)
CALL SET1 ; SEND REMAINING COMMANDS
CALL RDC1 ; READ IN ONE SECTOR
ORA A ; TEST FOR ERROR
JNZ BERR ; IF ERROR, ISSUE MESSAGE
INX D ; INCREMENT DISC ADDRESS
DEC C ; COUNT DOWN SECTORS
JNZ BT1 ; LOOP UNTIL DONE
JMP GOCMP ; SETUP AND RE-ENTER CP/M

BERR: CALL BTERR ; ISSUE ERROR MESSAGE
HLT ; HALT SYSTEM

BTERR: LXI H,BEMSG ; POINT TO ERROR MESSAGE
; ---- MESSAGE PRINTOUT ROUTINE ----

PTMSG: MOV A,M ; GET MESSAGE BYTE
CP1 '$' ; IS IT THE TERMINAL CHARACTER
NZ ; YES, SO RETURN
MOV C,A ; SAVE FOR CONSOLE OUTPUT
PUSH H
CALL CONOUT
POP H
INX H
JMP PTMSG

; ---- CORVUS DISC READ ROUTINE ----

READC: MV1 A,RDCOM ; GET READ COMMAND
CALL SETUP ; COMPUTE DISC ADDRESS AND ISSUCE COMMANDS
LHLD DMAAD ; GET DMA ADDRESS
RDCI: CALL TURN ; WAIT FOR ACCEPTANCE OF COMMAND
      JNZ ERRCD ; IF ERROR
      MVI B,SSIZE ; GET SECTOR SIZE
RLP:   IN STAT ; GET DRIVE STATUS
      ANI DRDY ; LOOK AT READY BIT
      JNZ RLP ; LOOP UNTIL BYTE IS AVAILABLE
      IN DATA ; READ BYTE FROM CONTROLLER
      MOV H,A ; SAVE IT IN MEMORY
      INX H
      DCR B ; COUNT DOWN BYTES
      JNZ RLP ; LOOP UNTIL DONE
RTN:   XRA A ; CLEAR ERROR INDICATOR
      RET

; ---- CORVUS DISC WRITE ROUTINE ----

WRITEC: MVI A,WRCON ; GET WRITE COMMAND
      CALL SETUP ; COMPUTE ADDRESS AND ISSUE COMMANDS
      MVI B,SSIZE ; GET SECTOR SIZE
      LHLD DMAAD ; GET DMA ADDRESS
WLP:    IN STAT ; GET DRIVE STATUS
      ANI DRDY ; LOOK AT READY BIT
      JNZ WLP ; LOOP UNTIL BYTE IS AVAILABLE
      MOV A,M ; GET BYTE FROM MEMORY
      OUT DATA ; SEND IT TO CONTROLLER
      INX H
      DCR B ; COUNT DOWN # OF BYTES
      JNZ WLP ; LOOP UNTIL DONE
      CALL TURN ; WAIT FOR BUSX TURN AROUND AND READ ERROR #
      JZ RTN ; RETURN IF OK
ERRCD:  PUSH B ; IF ERROR, ISSUE ERROR MESSAGE
      LXI H,ERMSG
      CALL PTMSG
      POP PSM ; GET ERROR # BACK IN ACC
      CALL HEXT ; PRINT IT OUT IN HEX
      LXI H,ERMSG1
      CALL PTMSG ; PRINT REMAINDER OF MESSAGE
      MVI A,1 ; SET ERROR INDICATOR
      RET

TURN:   IN STAT ; READ STATUS BYTE
      ANI DIFAC ; LOOK AT DRIVE ACTIVE BIT
      JNZ TURN
      CALL DELAY1 ; WAIT FOR OVER 20USEC
      CALL WAITI ; READ ERROR BYTE
      MOV B,A ; SAVE IT
      ANI SOH ; LOOK AT FATAL ERROR BIT
      RET

DELAY1: MVI B,6 ; DELAY MORE THAN 20USEC
DELAY:  DCR B ; COUNT DOWN
      JNZ DELAY
      RET

WAITI: IN STAT ; GET STATUS BYTE
ANI DRDY ; LOOK AT READY BIT
JNZ WAITI
IN DATA ; GET DATA FROM CONTROLLER
RET

; WAITO:
MOV B,A ; SAVE COMMAND
IN STAT ; READ STATUS BYTE
ANI DRDY ; LOOK AT READY BIT
JNZ WAITO+1
MOV A,B ; GET COMMAND
OUT DATA ; SEND IT TO CONTROLLER
RET

; --- OUTPUT ACC IN HEX ---

HEXOT: PUSH PSW ; SAVE BYTE
RRC ; SHIFT UPPER NIBBLE DOWN 4 BITS
RRC
RRC
RRC
CALL HEXB ; OUTPUT UPPER NIBBLE IN HEX
POP PSW ; RESTORE BYTE

HEXB: ANI OFH ; MASK OUT UPPER NIBBLE
ADI '0' ; ADD ASCII BIAS
CPI '9'+1 ; IS IT NUMERIC?
JNC HEXB ; YES, SO SEND IT OUT
ADI 7 ; NO, SO ADJUST FOR A-F
PRT: MOV C,A ; SAVE FOR OUTPUT
JMP CONOUT ; OUTPUT TO CONSOLE

; --- COMPUTE CORVUS DISC ADDRESS AND SEND TO CONTROLLER ---

SETUP: CALL WAITO ; ISSUE DISC R/W COMMAND
LHLD TRACK ; GET TRACK # FROM BUFFER
XCHG ; PUT IN (D,E)
LXI H,0 ; CLEAR CONVERSION BUFFER
LDA NSPTRK ; GET # SECTORS/TRACK (ASSUMED <255)
MVI B,B ; SET TO MULTIPLY 8 BITS
; MULTIPLY : (H,L)=TRACK* (# SECTORS/TRACK)
MULT: DAD H ; SHIFT BUFFER OVER 1 POSITION
RAL ; TEST NEXT BIT OF (#SECTORS/TRACK)
JNC MLI ; IF NOT A 1, DON'T ADD IN
DAC D ; IF A 1, ADD IN TRACK #
MLI: DCR B ; COUNT DOWN # BITS
JNZ MULT ; LOOP UNTIL DONE
XCHG ; PUT RESULT IN (D,E)
LHLD SECTOR ; GET SECTOR #
DAD D ; (H,L)=SECTOR+TRACK* (#SECTORS/TRACK)

; XCHG ; PUT RESULT IN (D,E)
LHLD ADDOF ; GET POINTER TO DISC ADDRESS OFFSET
; ADD IN DISC ADDRESS OFFSET
MOV A,E ; GET LOWER BYTE OF RELATIVE DISC ADDRESS
ADD M ; ADD IN LOWER BYTE OF ABSOLUTE DISC OFFSET
MOV E,A ; SAVE RESULT
INX H ; POINT TO NEXT BYTE OF OFFSET
MOV A,D ; DO ADDITION AGAIN
ADC M
MOV D,A ; SAVE IT
INX H ; POINT TO LAST BYTE OF OFFSET
MVI A,0 ; CLEAR ACC WITHOUT CLEARING CARRY BIT
ADC M ; GET UPPER BYTE OF DISC ADDRESS
RLC
RLC
RLC
MOV C,A ; SAVE IT
LDA CRDRIVE ; GET CORVUS DRIVE # (1-4)
ADD C ; MERGE IN EXTENDED DISC ADDRESS BITS

WE NOW HAVE (D,E)=LOWER TWO BYTES OF DISC ADDRESS
ACC =EXTENDED DISC ADDRESS+DRIVE #

SET1: CALL WAITO ; SEND DRIVE # TO CONTROLLER
MOV A,E
CALL WAITO ; SEND LOWER DISC ADDRESS TO CONTROLLER
MOV A,D
JMP WAITO

HOMECE: LXI B,0 ; GET TRACK 0
JMP SETTRK

---- SELECT DISC ROUTINE ----
NOTE, THIS ROUTINE DOES A LOT OF EXTRA WORK SO
THAT SOME OF IT NEED NOT BE DONE FOR EACH DISC
READ/WRITE OPERATION. THE METHOD USED TO SWITCH
BETWEEN CORVUS AND FLOPPY DRIVES (PATCHING A JUMP
TABLE) IS MAINLY USED BECAUSE IT CONCENTRATES THE
SELECT FUNCTIONS ALL WITHIN THE SELSDK ROUTINE.

SELDSDK: MOV A,C ; GET CP/M DRIVE #
LXI H,DSKNO ; POINT TO BUFFER WITH LAST DRIVE #
CMP M ; ARE THEY THE SAME?
JZ SLD3 ; YES, SO JUST GET POINTER AND RETURN
CPI DMAX ; NO, SO SEE IF # IS TOO BIG
JNC SLDERR ; ERROR, SO GIVE NOTICE
MOV M,C ; UPDATE DRIVE #
CPI MNPSUDO ; IS IT A FLOPPY?
JNC SLD1 ; YES, SO PROCESS SELECT

COPY CORVUS ROUTINE ADDRESSES INTO JUMP TABLE
LXI H,READC
SMLD DREAD+1
LXI H,WRITEC
SMLD DWRITE+1
LXI H,HOMEC
SMLD DHOME+1

MOV L,C ; GET CP/M DRIVE # IN (H,L)
MVI H,0
DAD H ; MULTIPLY BY 4
DAD H
LXI D,OFBSBAS ; POINT TO BASE OF OFFSET TABLE
FILE: BIOSC ASM PAGE 008

DAD D ; SELECT THE RIGHT ONE
SHLD ADDOF ; SAVE POINTER FOR LATER USE
INX H
INX H
INX H
MOV A,M ; GET ACTUAL CORVUS DRIVE #
STA CDRIVE ; SAVE IT
JMP SLD2 ; COMPUTE ADDRESS OF PARAM. BLOCK

COPY FLOPPY ROUTINE ADDRESSES INTO JUMP TABLE

SLD1: LXI H,READF
SHLD DREAD+1
LXI H,WRITEF
SHLD DWRITE+1
LXI H,HOMEF
SHLD DHOME+1

; PUSH B
CALL SELDF ; CALL FLOPPY SELECT ROUTINE
POP B

SLD2: MOV L,C ; GET CP/M DRIVE # IN (H,L)
MVI H,0
DAD H ; MULTIPLY BY 16
DAD H
DAD H
DAD H
DAD H
LXI D,DPBASE ; GET START OF PARAM. BLOCK
DAD D ; SELECT THE RIGHT BLOCK
SHLD PPOINT ; SAVE POINTER
LXI D,10
DAD D ; POINT TO ADDRESS OF DISC BLOCK
MOV E,M ; GET ADDRESS IN FROM TABLE INTO (D,E)
INX H
MOV D,M
XCHG
MOV E,M ; GET # SECTORS/TRACK INTO (D,E)
INX H
MOV D,M
XCHG
SHLD NSPTRK ; SAVE IT IN BUFFER

SLD3: LHL D PPOINT ; GET PARAM. POINTER
RET

SLDERR: LXI H,0 ; IF SELECT ERROR, GET 0 IN (H,L)
XRA A
STA CDISC ; SET TO REBOOT ON DRIVE A
RET

SETTRK: MOV L,C ; SAVE TRACK #
MOV H,B
SHLD TRACK
RET

; SETSEC: MOV L,C ; SAVE CP/M SECTOR #
MOV H,B
SHLD SECTOR
RET
FILE: BIOSC  ASM  PAGE 009

SETDMA: MOV  L,C ; SAVE DMA ADDRESS
         MOV  H,B
         SHLD  DMAAD
         RET

; ----- CONSOLE INPUT ROUTINE -----  
; (EXAMPLE, SIMPLE I/O PORT ORIENTED)
; CONIN: CALL  CONST ; CHECK CONSOLE STATUS
         ORA  A
         JZ  CONIN ; LOOP UNTIL READY
         IN  A ; GET CHAR.
         ANI  7FH ; MASK OFF PARITY
         RET

; ----- CONSOLE STATUS TEST -----  
; CONST: IN 0 ; GET STATUS
         ANI 20H ; MASK IT
         MVI  A,0 ; GET NOT READY INDICATOR
         RZ
         CMA ; RETURN WITH OFFH IF READY
         RET

; ----- CONSOLE OUTPUT -----  
; CONOUT: IN 0 ; GET STATUS BYTE
         ANI 2 ; MASK IT
         JZ  CONOUT ; LOOP UNTIL READY
         MOV  A,C
         OUT 1 ; OUTPUT TO CONSOLE
         RET

; ----- LIST DEVICE DRIVERS -----  
; LIST:  NOP ; PUT IN CODE FOR LIST DEVICE
         RET

; ----- LIST STATUS TEST -----  
; LISTST: XRA  A ; CLEAR STATUS
         RET

; ----- PUNCH DEVICE -----  
; PUNCH:  NOP ; PUT IN CODE FOR PUNCH
         RET

; ----- READER DEVICE -----  
; READER: MVI  A, 'Z' - 40H ; RETURN CONTROL-Z
         RET

; ------------------- FLOPPY DISC ROUTINES -------------------  
; (USED TRACK, SECTOR, AND DMA ADDRESS IN BUFFERS)

; ----- READ SECTOR FROM FLOPPY -----
FILE: BIOSC ASM PAGE 010

; READF: NOP ; PUT IN CODE FOR READ ROUTINE
; XRA A ; CLEAR FLAGS
; RET

; ---- WRITE SECTOR TO FLOPPY ----
; WRITF: NOP ; PUT IN CODE FOR WRITE ROUTINE
; XRA A
; RET

; ---- HOME THE FLOPPY ----
; HOMEF: NOP ; PUT IN CODE FOR HOME ROUTINE
; RET

; ---- SELECT FLOPPY ----
; SELDF: NOP ; PUT IN CODE TO SELECT BETWEEN FLOPPYS
; RET

; ---------- MESSAGES ----------
; BMSG: DB ODH,0AH,’ ----- CORVUS ’
; DB MSIZE/10*’0’, MSIZE MOD 10 + ’0’
; DB ’K CP/M V2.0 OF 2-26-80 -----’, ODH,0AH,’$’

; EMSG: DB ODH,0AH,07,’ ** BOOT ERROR **’,ODH,0AH,’$’

; ERMSG: DB ODH,0AH,07,’ -- DISC R/W ERROR # $’

; ERMSG1: DB ’H --’,ODH,0AH,’$’

;---------- BUFFERS ----------
; DMAAD: DS 2 ; DMA ADDRESS
; TRACK: DS 2 ; TRACK #
; SECTOR: DS 2 ; SECTOR #
; DSKNO: DB OFFH ; CURRENT DISC # (UNDEFINED AT START)
; ADDOF: DS 2 ; BUFFER FOR POINTER TO ADDRESS OFFSET
; NSPTRK: DS 2 ; BUFFER WITH # SECTORS/TRACK
; PPINT: DS 2 ; POINTER TO CURRENT PARAM. BLOCK
; CDRIVE: DS 1 ; BUFFER FOR CORVUS DISC #
; DIRBUF: DS 128 ; DIRECTORY ACCESS BUFFER
; ALVO: DS 74 ; DRIVE 0 ALLOC. MAP
; CSV0: DS 0 ; DRIVE 0 CHECK BUFFER (NOT USED)
; ALV1: DS 74 ; DRIVE 1 ALLOC. MAP
; CSV1: DS 0 ; DRIVE 1 CHECK BUFFER
; ALV2: DS 31 ; DRIVE 2 ALLOC. MAP (FLOPPY)
; CSV2: DS 16 ; CHECKSUM ARRAY
; ALV3: DS 31 ; DRIVE 3 ALLOC. MAP
; CSV3: DS 16 ; CHECKSUM ARRAY

; END
FILE: BIOSCT  ASM  PAGE 001

; CORVUS DISC DRIVERS FOR CP/M 2.0  (BIOS)
; ----- WITH TARBELL 8 INCH FLOPPY DRIVERS -----
; VERSION 1.21T
; BY BRK
;
; MSIZE EQU 20   ; CP/M VERSION MEMORY SIZE IN KB
;
; DELTA EQU 0000H ; OFFSET FROM STD CP/M SIZE
; BIAS EQU (MSIZE-20)*1024-DELTA ; OFFSET FROM 20K CP/M
; CCP EQU 3400H+BIAS* ; BASE OF CP/M
;
; OFFSET EQU 980H-CCP ; OFFSET USED WITH DDT IN
; SYSTEM CONFIGURATION
;
; BDOS EQU CCP+806H ; BASE OF BDOS
; BIOS EQU CCP+1600H ; BASE OF BIOS
; NSSEC EQU (BIOS-CCP)/128 ; # SECTORS TO BOOT
; CUDISC EQU 04 ; BUFFER LOCATION FOR CURRENT DISC #
; IOBYTE EQU 03 ; LOCATION OF INTEL IOBYTE
;
; ----- CORVUS EQUATES ----- 
;
; DATA EQU ODEH ; DISC I/O PORT #
; STAT EQU DATA+1 ; DISC STATUS PORT
; DRYD EQU 1 ; MASK FOR DRIVE READY BIT
; DIFAC EQU 2 ; MASK FOR DRIVE ACTIVE BIT
; RDCOM EQU 12H ; READ COMMAND (VERS. 1 CCODE)
; WRCOM EQU 13H ; WRITE COMMAND (VERS. 1 CCODE)
; NSPSVD EQU 2 ; NUMBER OF PSEUDO DRIVES ON SINGLE CORVUS DRIVE
; DMAX EQU 4 ; TOTAL # OF DRIVES (INCLUDES TWO 8 INCH FLOPPIES)
; SSIZE EQU 128 ; SECTOR SIZE (IN BYTES)
; BDREDE EQU 1 ; CORVUS DRIVE # TO BOOT FROM
; CSEC EQU 13 ; STARTING DISC ADDRESS FOR CP/M BOOT
;
;    ORG BIOS

;  CP/M INTERFACE JUMP TABLE
;
; JMP     BDREDE
; WBOOT   JMP     WBOOT
; CONST   JMP     CONST
; CONIN   JMP     CONIN
; CONOUT  JMP     CONOUT
; LIST    JMP     LIST
; PUNCH   JMP     PUNCH
; READER  JMP     READER
; DHOME   JMP     DHOME
; SLEDSK  JMP     SLEDSK
; SETTRK  JMP     SETTRK
; SETSEC  JMP     SETSEC
; SETDMA  JMP     SETDMA
; DREAD   JMP     DREAD
; DWRT    JMP     DWRT
FILE: BIOSCT  ASM    PAGE 002

JMP LISTST ; LIST DEVICE STATUS REQUEST
JMP SECTRAN ; SECTOR TRANSLATION ROUTINE

--- DISK PARAMETER BLOCKS ---
THE EXAMPLE HERE DIVIDES ONE 9.7MBYTE CORVUS DISC INTO
TWO LARGE PSEUDO DRIVES (OF EQUAL SIZE)
AND ALSO PROVIDES FOR THE INTERFACE OF TWO STANDARD 8 INCH
SINGLE DENSITY FLOPPY DISC DRIVES.

NOTE:
THE NUMBERS SHOWN IN DPBC (THE PARAMETER BLOCK)
FOR THE PSEUDO DRIVE AND ITS ASSOCIATED ALLOCATION
BUFFER SIZES ARE THE RESULT OF CHOOSING:
37860 SECTORS/PSEUDO DRIVE
60 SECTORS/TRACK
1 RESERVED TRACK FOR OPERATING SYSTEM
256 DIRECTORY ENTRIES
8*1024 BYTE BLOCKS

DPBASE EQU $  

DPE0: DW 0,0 ; CORVUS PSEUDO DRIVE 1
DW 0,0
DW DIRBUF,DPBC ; DIRECTORY BUFFER, PARAM. BLOCK
DW CSV0,ALVO ; CHECK, ALLOC MAP

DPE1: DW 0,0 ; CORVUS PSEUDO DRIVE 2
DW 0,0
DW DIRBUF,DPBC ; DIRECTORY BUFFER, PARAM. BLOCK
DW CSV1,ALV1 ; CHECK, ALLOC MAP

DPE2: DW FTAB,0 ; FLOPPY TRANSLATION TABLE
DW 0,0
DW DIRBUF,DPBF ; DIRECTORY BUFFER, PARAM. BLOCK
DW CSV2,ALV2 ; CHECK, ALLOC MAP

DPE3: DW FTAB,0 ; FLOPPY TRANSLATION TABLE
DW 0,0
DW DIRBUF,DPBF ; DIRECTORY BUFFER, PARAM. BLOCK
DW CSV3,ALV3 ; CHECK, ALLOC MAP

DPBC: DW 60 ; SECTORS/TRACK ON CORVUS PSEUDO DRIVE
DB 6 ; BLOCK SHIFT
DB 63 ; BLOCK MASK
DB 3 ; EXTENT MASK
DW 589 ; DISK SIZE-1
DW 255 ; DIRECTORY MAX
DB 128 ; ALLOC0
DB 0 ; ALLOCI
DW 0 ; CHECK SIZE
DW 1 ; OFFSET

DPBF: DW 26 ; SECTORS/TRACK ON STD 8 INCH FLOPPY
DB 3 ; BLOCK SHIFT FACTOR
DB 7 ; BLOCK MASK
DB 0 ; NULL MASK
DW 242 ; DISK SIZE-1
DW 63 ; DIRECTORY MAX
FILE: BIOSCT  ASM  PAGE 003

DB  192  ; ALLOC 0
DB  0  ; ALLOC 1
DW  16  ; CHECK SIZE
DW  2  ; TRACK OFFSET

; --- CORVUS DISC OFFSET TABLE ---

OFSBAS EQU $  ; STARTING DISC ADDRESS FOR DRIVE 0
PDRVO: DW CSEC-1  ; THIS IS THE UPPER BYTE OF THE 20 BIT DISC ADDRESS
         DB 0  ; ACTUAL PHYSICAL DRIVE # (1-4)

PDRVI: DW 37884  ; STARTING DISC ADDRESS FOR DRIVE 1
          DB 0  ; ACTUAL PHYSICAL DRIVE # (1-4)

; --- STANDARD 8 INCH FLOPPY INTERLACE TABLE ---

FTAB: DB 1,7,13,19
       DB 25,5,11,17
       DB 23,3,9,15
       DB 21,2,8,14
       DB 20,26,6,12
       DB 18,24,4,10
       DB 16,22

; --- AUXILIARY JUMP TABLE FOR DRIVE SWITCHING ---

DHOME: JMP HOME  ; SET TO HOME CORVUS DRIVE
DREAD: JMP READC  ; SET TO READ FROM CORVUS DRIVE
DWRITE: JMP WRITEC  ; SET TO WRITE TO CORVUS DRIVE

; --- SECTOR TRANSLATION ROUTINE ---

SECTRN: MOV A,D  ; TEST IF TABLE TRANSLATION IS REQUESTED
         ORA E
         JNZ STRI  ; YES, SO DO IT
         MOV L,C
         MOV H,B
         RET

STRI: XCHG B  ; GET TABLE ADDRESS IN (H,L)
       ADD L  ; INDEX INTO TABLE
       MOV L,M
       MOV H,0
       RET

; --- COLD BOOT STARTUP ---

BOOT: LXI SP,80H  ; SETUP TEMP. STACK
       LXI H,BMSG  ; POINT TO BOOT UP MESSAGE
       CALL PTMSG  ; PRINT IT OUT
       MVI A,0  ; GET CURRENT DISC #
       STA CDISC  ; SAVE IN BUFFER
       MOV C,A
       CALL SELDSK  ; SELECT IT ALSO (INITIALIZE BUFFERS)

GOCALM: MVI A,0C3H  ; GET JUMP INSTRUCTION
         STA 0  ; SETUP FOR WARM BOOT
FILE: BIOSCT  ASM  PAGE 004

LXI  H,WBOOTE ; WARM BOOT ENTRY
SHLD 1 ; SET ADDRESS
STA  5 ; SETUP BDOS ENTRY JUMP
LXI  H,BDOS
SHLD 6
LXI B,80H ; DEFAULT DMA ADDRESS
CALL SETDMA
LDA CDISC ; GET CURRENT DRIVE #
MOV C,A ; SAVEZ FOR CCP FUNCTION
JMP CCP ; ENTER CP/M

; ---- WARM BOOT STARTUP ROUTINE ----

WBOOT: LXI SP,80H ; SET STACK
WBO1: MVI A,OFFH ; GET INVALID COMMAND
        OUT DATA ; SEND IT TO CONTROLLER
        MVI B,150 ; SET FOR LONG DELAY
        CALL DELAY
        IN STAT ; GET STATUS BYTE
        ANI DIFAC ; LOOK AT DRIVE ACTIVE BIT
        JNZ WBO1 ; LOOP UNTIL NOT ACTIVE
        CALL WAIT1 ; WAIT FOR ERROR CODE
        CPI 8FH ; CHECK RETURN CODE
        JNZ WBO1 ; IF NOT RIGHT, TRY AGAIN

        MVI C,NSEC ; GET # SECTORS TO BOOT
        LXI H,CMP ; GET RAM START ADDRESS OF LOAD
        LXI D,CSEC ; GET DISC ADDRESS FOR COPY OF CP/M

BTI:  MVI A,RDCOM ; GET READ COMMAND
        CALL WAIT0 ; SEND IT TO CONTROLLER
        MVI A,BDRIVE ; GET BOOTUP DRIVE # (CORVUS PHYSICAL DRIVE)
        CALL SETI ; SEND REMAINING COMMANDS
        CALL RDC1 ; READ IN ONE SECTOR
        ORA A ; TEST FOR ERROR
        JNZ BERR ; IF ERROR, ISSUE MESSAGE
        INX D ; INCREMENT DISC ADDRESS
        DCR C ; COUNT DOWN SECTORS
        JNZ BTI ; LOOP UNTIL DONE
        JMP GOCPM ; SETUP AND RE-ENTER CP/M

BERR: CALL BTERR ; ISSUE ERROR MESSAGE
        HALT ; HALT SYSTEM

BTERR: LXI H,BEMSG ; POINT TO ERROR MESSAGE

; ---- MESSAGE PRINTOUT ROUTINE ----

PTMSG: MOV A,M ; GET MESSAGE BYTE
        CPI "$" ; IS IT THE TERMINAL CHARACTER
        RZ ; YES, SO RETURN
        MOV C,A ; SAVE FOR CONSOLE OUTPUT
        PUSH H
        CALL CONOUT
        POP H
        INX H
        JMP PTMSG
FILE: BIOSCTR ASH PAGE 005

; --- CORVUS DISC READ ROUTINE ---

READC:  MVI A,ROCOM ; GET READ COMMAND
CALL SETUP ; COMPUTE DISC ADDRESS AND ISSUE COMMANDS
LHLD DMAAD ; GET DMA ADDRESS

RDG:   CALL TURN ; WAIT FOR ACCEPTANCE OF COMMAND
JNZ ERRCD ; IF ERROR
MVI B,SIZE ; GET SECTOR SIZE

RLP:    IN STAT ; GET DRIVE STATUS
ANI DRDY ; LOOK AT READY BIT
JNZ RLP ; LOOP UNTIL BYTE IS AVAILABLE
IN DATA ; READ BYTE FROM CONTROLLER
MOV H,A ; SAVE IT IN MEMORY
INX H
DCR B ; COUNT DOWNBYTES
JNZ RLP ; LOOP UNTIL DONE

RTN:    XRA A ; CLEAR ERROR INDICATOR
RET

; --- CORVUS DISC WRITE ROUTINE ---

WRITEC: MVI A,WRCOM ; GET WRITE COMMAND
CALL SETUP ; COMPUTE ADDRESS AND ISSUE COMMANDS
MVI B,SIZE ; GET SECTOR SIZE
LHLD DMAAD ; GET DMA ADDRESS

WLP:    IN STAT ; GET DRIVE STATUS
ANI DRDY ; LOOK AT READY BIT
JNZ WLP ; LOOP UNTIL BYTE IS AVAILABLE
MOV A,M ; GET BYTE FROM MEMORY
OUT DATA ; SEND IT TO CONTROLLER
INX H
DCR B ; COUNT DOWN # OFBYTES
JNZ WLP ; LOOP UNTIL DONE
CALL TURN ; WAIT FOR BUSB TURN AROUND AND READ ERROR #
JZ RTN ; RETURN IF OK

ERRCD:  PUSH B ; IF ERROR, ISSUE ERROR MESSAGE
LXI H,ERMSG
CALL PTMSG
POP PSW ; GET ERROR # BACK IN ACC
CALL HEXOT ; PRINT IT OUT IN HEX
LXI H,ERMSG1
CALL PTMSG ; PRINT REMAINDER OF MESSAGE
MVI A,1 ; SET ERROR INDICATOR
RET

TURN:   IN STAT ; READ STATUS BYTE
ANI DIFAC ; LOOK AT DRIVE ACTIVE BIT
JNZ TURN
CALL DELAY1 ; WAIT FOR OVER 20USEC
CALL WAITI ; READ ERROR BYTE
MOV B,A ; SAVE IT
ANI BON ; LOOK AT BAD ERROR BIT
RET

DELAY1: MVI 8,6 ; DELAY MORE THAN 20USEC
DELAY:  DCR B ; COUNT DOWN
JNZ DELAY
RET

; WAITI: IN STAT ; GET STATUS BYTE
AMI DRDY ; LOOK AT READY BIT
JNZ WAITI
IN DATA ; GET DATA FROM CONTROLLER
RET

; WAITO: MOV B,A ; SAVE COMMAND
IN STAT ; READ STATUS BYTE
AMI DRDY ; LOOK AT READY BIT
JNZ WAITO+1
MOV A,B ; GET COMMAND
OUT DATA ; SEND IT TO CONTROLLER
RET

; --- OUTPUT ACC IN HEX ---

HEXO: PUSH PSW ; SAVE BYTE
RRC PSW ; SHIFT UPPER NIBBLE DOWN 4 BITS
RRC
RRC
RRC
CALL HEXB ; OUTPUT UPPER NIBBLE IN HEX
POP PSW ; RESTORE BYTE
HEXB: AMI OFH ; MASK OUT UPPER NIBBLE
ADI '0' ; ADD ASCII BIAS
CPI '9'+1 ; IS IT NUMERIC?
JC PRT ; YES, SO SEND IT OUT
ADI 7 ; NO, SO ADJUST FOR A-F
PRT: MOV C,A ; SAVE FOR OUTPUT
JMP CONOUT ; OUTPUT TO CONSOLE

; --- COMPUTE CORVUS DISC ADDRESS AND SEND TO CONTROLLER ---

SETUP: CALL WAITO ; ISSUE DISC R/W COMMAND
LHLD TRACK ; GET TRACK # FROM BUFFER
XCHG ; PUT IN (D,E)
LXI H,0 ; CLEAR CONVERSION BUFFER
LDA NSPTRK ; GET # SECTORS/TRACK (ASSUMED <255)
MVI B,8 ; SET TO MULTIPLY 8 BITS
; MULTIPLY : (H,L)=TRACK (# SECTORS/TRACK)
MUL:
DAD H ; SHIFT BUFFER OVER 1 POSITION
RAL ; TEST NEXT BIT OF (#SECTORS/TRACK)
JNC ML1 ; IF NOT A 1, DON'T ADD IN
DAD D ; IF A 1, ADD IN TRACK #
ML1:
DCR B ; COUNT DOWN # BITS
JNZ MULT ; LOOP UNTIL DONE
XCHG ; PUT RESULT IN (D,E)
LHLD SECTOR ; GET SECTOR #
DAD D ; (H,L)=SECTOR+TRACK (#SECTORS/TRACK)
; XCHG ; PUT RESULT IN (D,E)
LHLD ADDOF ; GET POINTER TO DISC ADDRESS OFFSET
; ADD IN DISC ADDRESS OFFSET
MOV A,E ; GET LOWER BYTE OF RELATIVE DISC ADDRESS
ADD H ; ADD IN LOWER BYTE OF ABSOLUTE DISC OFFSET
MOV E,A ; SAVE RESULT
INX H ; POINT TO NEXT BYTE OF OFFSET
MOV A,D ; DO ADDITION AGAIN
ADC M
MOV D,A ; SAVE IT
INX H ; POINT TO LAST BYTE OF OFFSET
MVI A,O ; CLEAR ACC WITHOUT CLEARING CARRY BIT
ADC M ; GET UPPER BYTE OF DISC ADDRESS
RLC
RLC
RLC
MOV C,A ; SAVE IT
LDA CDRIVE ; GET CORVUS DRIVE # (1-4)
ADD C ; MERGE IN EXTENDED DISC ADDRESS BITS

; WE NOW HAVE (D,E)=LOWER TWO BYTES OF DISC ADDRESS
; ACC = EXTENDED DISC ADDRESS+DRIVE #

STDI: CALL WAITO ; SEND DRIVE # TO CONTROLLER
MOV A,E
CALL WAITO ; SEND LOWER DISC ADDRESS TO CONTROLLER
MOV A,D
JMP WAITO

; --- HOME CORVUS DRIVE -----
HOMEC: LXI H,0 ; GET TRACK 0
JMP SETTRK

; --- SELECT DISC ROUTINE ---
; NOTE: THIS ROUTINE DOES A LOT OF EXTRA WORK SO
; THAT SOME OF IT NEED NOT BE DONE FOR EACH DISC
; READ/WRITE OPERATION. THE METHOD USED TO SWITCH
; BETWEEN CORVUS AND FLOPPY DRIVES (PATCHING A JUMP
; TABLE) IS MAINLY USED BECAUSE IT CONCENTRATES THE
; SELECT FUNCTIONS ALL WITHIN THE SELSDK ROUTINE.

SELDSDK: MOV A,C ; GET CP/M DRIVE #
LXI H,DSKNO ; POINT TO BUFFER WITH LAST DRIVE #
CMP M ; ARE THEY THE SAME?
JZ SLD3 ; YES, SO JUST GET POINTER AND RETURN
CPI DMAX ; NO, SO SEE IF # IS TOO BIG
JNC SLDERR ; ERROR, SO GIVE NOTICE
MOV M,C ; UPDATE DRIVE #
CPI NPSUDO ; IS IT A FLOPPY?
JNC SLD1 ; YES, SO PROCESS SELECT

COPY CORVUS ROUTINE ADDRESSES INTO JUMP TABLE
LXI H,READC
SULD DREAD+1
LXI H,WRITEC
SULD DWRITE+1
LXI H,HOMEC
SULD DHOME+1

MOV I,C ; GET CP/M DRIVE # IN (H,L)
MV1 H,0
DAD  H       ; MULTIPy BY 4
DAD  H
LXI  H, OFSBAS ; POINT TO BASE OF OFFSET TABLE
DAD  D       ; SELECT THE RIGHT ONE
SHLD ADDOF  ; SAVE POINTER FOR LATER USE
INX  H
INX  H
INX  H
MOV  A,M    ; GET ACTUAL CORVUS DRIVE #
STA  CDRIEf ; SAVE IT
JMP  SLD2    ; COMPUTE ADDRESS OF PARAM. BLOCK

; COPY FLOPPY ROUTINE ADDRESSES INTO JUMP TABLE
SLD1: LXI  H, READF
SHLD DREADF+1
LXI  H, WRITEF
SHLD DWRT+1
LXI  H, HOMEF
SHLD DHOME+1

; PUSH B
CALL SELDF ; CALL FLOPPY SELECT ROUTINE
POP  B

SLD2: MOV  L,C    ; GET CP/M DRIVE # IN (H,L)
MVI  H, 0
DAD  H       ; MULTIPLY BY 16
DAD  H
DAD  H
LXI  D, DFRASE ; GET START OF PARAM. BLOCK
DAD  D       ; SELECT THE RIGHT BLOCK
SHLD PPPOINT ; SAVE POINTER
LXI  D, 10
DAD  D       ; POINT TO ADDRESS OF DISC BLOCK
MOV  E,M    ; GET ADDRESS IN FROM TABLE INTO (D,E)
INX  H
MOV  D, M
XCHG
MOV  E,M    ; PUT IN (H,L)
MOV  D, M    ; GET # SECTORS/TRACK INTO (D,E)
INX  H
MOV  D, M
XCHG
SHLD NSPTRK ; SAVE IT IN BUFFER
SLD3: LHLd  PPPOINT ; GET PARAM. POINTER
RET
SLDERR: LXI  H, 0 ; IF SELECT ERROR, GET 0 IN (H,L)
XRA  A
STA  CDISC    ; SET TO REBOOT ON DRIVE A
RET

; SETTRK: MOV  L, C ; SAVE TRACK #
MOV  H, R
SHLD  TRACK
RET

; SETSEC: MOV  L, C ; SAVE CP/M SECTOR #
MOV  H, R
SHLD SECTOR
RET ;

SETDMA: MOV L,C ; SAVE DMA ADDRESS
MOV H,B
SHLD DMAAD
RET ;

;----- CONSOLE INPUT ROUTINE -----
; (EXAMPLE, SIMPLE I/O PORT ORIENTED)
;
CONIN: CALL CONST ; CHECK CONSOLE STATUS
ORA A
JZ CONIN ; LOOP UNTIL READY
IN 1 ; GET CHAR.
ANI 7FH ; MASK OFF PARITY
RET ;

;----- CONSOLE STATUS TEST -----;
;
CONST: IN 0 ; GET STATUS
ANI 20H ; MASK IT
MVI A,0 ; GET NOT READY INDICATOR
RZ
CMA ; RETURN WITH OFFH IF READY
RET ;

;----- CONSOLE OUTPUT -----;

CONOUT: IN 0 ; GET STATUS BYTE
ANI 2 ; MASK IT
JZ CONOUT ; LOOP UNTIL READY
MOV A,C
OUT 1 ; OUTPUT TO CONSOLE
RET ;

;----- LIST DEVICE DRIVERS -----;

LIST: NOP ; PUT IN CODE FOR LIST DEVICE
RET ;

;----- LIST STATUS TEST -----
;
LISTST: XRA A ; CLEAR STATUS
RET ;

;----- PUNCH DEVICE -----;

PUNCH: NOP ; PUT IN CODE FOR PUNCH
RET ;

;----- READER DEVICE -----;

READER: MVI A, 'Z'-40H ; RETURN CONTROL-Z
RET ;

;---------------- FLOPPY DISC ROUTINES -----------------
; (USE TRACK, SECTOR, AND DMA ADDRESS IN BUFFERS)
; ----- DISC CONTROL ROUTINES FOR TARBEILL 8 INCH FLOPPYS ------
DCOM EQU OFBH ; START OF TARBEILL I/O PORTS
DSAT EQU DCOM
DTRK EQU DCOM+1
SECTP EQU DCOM+2
DDATA EQU DCOM+3
WAIT EQU DCOM+4

; ------ MOVE HEAD TO TRACK ZERO -------
HOMEF MVI A,0DOH
OUT DCOM
HOMEF IN DSTAT
RRC
JC HOMEF
MVI A,3
OUT DCOM
IN WAIT
ORA A
MVI A,1
JM ERROR
IN DSTAT
MOV E,A
ANI 4
JZ HERR
MOV A,E
ANI 91H
RET

HERR MVI A,1
ORA A
RET

; ----- SELECT DISC NUMBER -------
SELDFF MOV A,C
LXI H,DSKNO1
CHP M
RZ
MOV M,A ; UPDATE ACTIVE DRIVE #
CPI NPSUDO ; TEST FOR FIRST FLOPPY
LXI H,TMK1 ; POINT TO HEAD BUFFER
IN DTRK ; GET CURRENT TRACK
PUSH PSW ; SAVE IT
MOV A,M ; GET OTHER HEAD POSITION
OUT DTRK ; SET INTO CONTROLLER
POP PSW ; GET CURRENT ONE BACK
MOV M,A ; SAVE IT
MVI A,0E2H ; COMMAND TO SET TO SECOND FLOPPY
JNZ DSK1
MVI A,0F2H ; COMMAND TO SET TO FIRST FLOPPY
DSK1 OUT WAIT
XRA A
RET
; ; ----- READ THE SECTOR SPECIFIED BY THE BUFFERS ----- 
; USE THE STARTING ADDRESS AT DMAAD
; 
; READF LDA TRACK
; CALL SEEK
; RNZ
; LDA SECTOR
; LHLD DMAAD
; READ1 OUT SECTP
; IN DSTAT
; RRC
; MVI A,1
; JC ERROR
; MVI A,8CH
; READE MVI C,10H
; ; SET # OF ERROR TRIALS
; OUT DCOM
; RLOOP IN WAIT
; ORA A
; JP RDONE
; IN DDATA
; MOV M,A
; INX H
; JMP RLOOP
; ; RDONE IN DSTAT
; AMI 9DH
; RZ
; ; RETURN IF OK
; DCR C
; ; DECREMENT ERROR COUNT
; JNZ READF
; JMP PROC R+1
; ; ----- WRITE THE SECTOR SPECIFIED BY THE BUFFERS ----- 
; USE STARTING ADDRESS AT DMAAD
; 
; WRITEF LDA TRACK
; CALL SEEK
; RNZ
; LDA SECTOR
; LHLD DMAAD
; OUT SECTP
; IN DSTAT
; RRC
; MVI A,1
; JC ERROR
; MVI A,0ACH
; OUT DCOM
; WLOOP IN WAIT
; ORA A
; JP WDONE
; MOV A,M
; OUT DDATA
; INX H
; JMP WLOOP
; ; WDONE IN DSTAT
ANI OFDH
PROC RZ
MOV B,A
ANI OCOH
JZ WERR1
MVI A,2
WERR1 ORA B
MOV B,A
ANI 4
RRC
RRC
ORA B
MOV B,A
ANI 20H
JZ WERR2
XRA A
OUT WAIT
INR A
WERR2 ORA B
ANI 1BH
ORA A
RET

; SEEK PUSH B
MOV B,A
IN DEKR
CMP B
MOV A,B
POP B
RZ
OUT DDATA
IN DSTAT
RRC
MVI A,1
JC ERROR
MVI A,13H
OUT DCOM
IN WAIT
ORA A
MVI A,1
JH ERROR
IN DSTAT
ANI 91H
RZ
JP ERROR
ANI 7FH
ORI 2
ERROR ORA A
RET

; ----- END OF FLOPPY CONTROL ROUTINES ---------

; ----- MESSAGES ---------

; BMSG:  DB ODH,OAH,'----- CORVUS
DB MSIZE/10+'0',MSIZE MOD 10 + '0'
DB 'K CP/H V2.0 OF 2-26-80 -----',ODH,OAH,'$'
FILE: BIOSCT ASM PAGE 013

; BEMSG: DB ODH,0AH,07, ' ** ROOT ERROR **',ODH,0AH, ' $' ;
; ERMG: DB ODH,0AH,07, ' -- DISC R/W ERROR # $'
; ERRSIG: DB 'H --',ODH,0AH, ' $'
;
; ---------- BUFFERS -----------
;
DHAAD: DS 2 ; DHA ADDRESS
TRACK: DS 2 ; TRACK #
SECTOR: DS 2 ; SECTOR #
DSKNO: DB OFFH ; CURRENT DISC # (UNDEFINED AT START)
USKNO1: DB OFFH ; CURRENT FLOPPY # (UNDEFINED AT START)
TRKL: DS 1 ; BUFFER FOR FLOPPY HEAD POSITION
ADDOF: DS 2 ; BUFFER FOR POINTER TO ADDRESS OFFSET
NSPTK: DS 2 ; BUFFER WITH # SECTORS/TRACK
POINTE: DS 2 ; POINTER TO CURRENT PARAM. BLOCK
CRDVE: DS 1 ; BUFFER FOR CORVUS DISC #
;
DIRBUF: DS 128 ; DIRECTORY ACCESS BUFFER
ALVO: DS 74 ; DRIVE 0 ALLOC. MAP
CSV0: DS 0 ; DRIVE 0 CHECK BUFFER (NOT USED)
ALV1: DS 74 ; DRIVE 1 ALLOC. MAP
CSV1: DS 0 ; DRIVE 1 CHECK BUFFER
ALV2: DS 31 ; DRIVE 2 ALLOC. MAP (FLOPPY)
CSV2: DS 16 ; CHECKSUM ARRAY
ALV3: DS 31 ; DRIVE 3 ALLOC. MAP
CSV3: DS 16 ; CHECKSUM ARRAY
;
END
FILE: CLINK2  ASM  PAGE 001

*******************************************************************************

CORVUS FLOPPY CP/M LINK PROGRAM
FOR CP/M 2.X

VERSION 1.02 BY BRK

COPYRIGHT 1980 CORVUS.  ALL RIGHTS RESERVED

*******************************************************************************

THIS PROGRAM IS DESIGNED TO LINK A FLOPPY DISC CP/M V2.X TO THE CORVUS HARD DISC. IF YOUR CP/M HAS TWO FLOPPYS AND YOU CHOOSE TO KEEP THE CORVUS DISC LAYOUT CHOSEN BY THIS PROGRAM, YOU NEED ONLY CHOOSE WHERE THIS PROGRAM IS TO LOAD (SEE THE EQUATE FOR FREE IN THIS PROGRAM).

THIS PROGRAM LOADS AT THE CP/M TPA (100H) AND MOVES A PART OF ITSELF UP TO LOCATION:FREE AND THEN PATCHES THE CODE IN THIS AREA TO LINK TO THE CURRENTLY RUNNING FLOPPY CP/M 2.X. THIS INVOLVES COPYING SOME TABLES FROM THE CP/M UP INTO THE LINK CODE AND RE-DIRECTING SOME JUMPS IN THE BIOS JUMP TABLE UP TO THE LINK CODE. THIS LINK WILL NOT BE BROKEN BY A WARM BOOT BUT A COLD BOOT WILL BRING IN A NEW COPY OF CP/M THAT IS NOT LINKED.

THIS PROGRAM MUST NOT BE RUN MORE THAN ONCE AFTER EACH COLD START SINCE IT WOULD TRY TO SORT OF LINK TO ITSELF IF RUN AGAIN (NOTE: THERE IS A BUILT IN CHECKING RUTINE TO PREVENT THIS LINK FROM BEING ENABLED TWICE- TO AVOID THIS PROBLEM). THIS PROGRAM IS AN IDEAL CANDIDATE FOR USING THE PROGRAM AUTO LOAD FEATURE BUILT INTO THE CCP OF CP/M 2.0. THIS FEATURE ALLOWS CP/M TO AUTOMATICALLY EXECUTE A PRE-SELECTED COMMAND (SUCH AS LOADING BASIC OR DOING A DIRECTORY LISTING) ON EACH WARM OR COLD BOOT.

THIS FEATURE CAN BE MADE SELECTIVE TO ONLY COLD BOOTS BY A MODIFICATION OF THE USER BIOS.

IN ANY CASE, YOU CAN ALWAYS DO THE LINK MANUALLY BY LOADING CLINK AFTER YOU FIRST BOOT UP. NOTE, A CLEANER SOLUTION IS TO CONFIGURE THESE DRIVERS DIRECTLY INTO YOUR CP/M (AS CAN BE DONE WITH THE FILE BIOSC.ASM) HOWEVER IF YOU DO NOT HAVE THE SOURCE FOR YOUR DRIVERS, THIS MAY NOT BE POSSIBLE.

*******************************************************************************

FMAX EQU 2 ;NUMBER OF FLOPPY DRIVES
DMAX EQU 4 ;NUMBER OF DRIVES (TOTAL)
FREE EQU 0D000H ;FREE AREA FOR THIS CODE TO
                ;BE MOVED TO ABOVE CP/M

*******************************************************************************
FILE: CLINK2 ASM PAGE 002

; ---- CP/M EQUATES ----
CDISC EQU 04 ; BUFFER LOCATION FOR CURRENT DISC #
I0BYTE EQU 03 ; LOCATION OF INTEL I0BYTE
BDOS EQU 05 ; LOCATION OF BDOS JUMP
RSET EQU 13 ; BDOS RESET COMMAND
LST EQU 9 ; LIST STRING COMMAND
CONOUT EQU 4A0CH ; LOCATION OF CONOUT JUMP IN 20K
; CP/M 2.0 BIOS (DO NOT CHANGE)

; ---- CORVUS EQUATES ----
DATA EQU ODH ; DISC I/O PORT #
STAT EQU DATA+1 ; DISC STATUS PORT
URDY EQU 1 ; MASK FOR DRIVE READY BIT
DIFAC EQU 2 ; MASK FOR DRIVE ACTIVE BIT
RDCOM EQU 12H ; READ COMMAND (VERS. 1 CODE)
WRCOM EQU 13H ; WRITE COMMAND (VERS. 1 CODE)
SSIZE EQU 128 ; SECTOR SIZE (IN BYTES)

ORG 100H ; START AT CP/M TPA

; --- COMPUTE ADDRESS OF BIOS FOR THE LOADED CP/M ---
LHLD I ; GET ADDRESS OF JUMP TO WBOOT
DCX H
DCX H
SHLD LDEL0S ; SAVE IT FOR LATER USE

; --- COMPUTE CP/M OFFSET BIAS FROM 20K SYSTEM ----
MOV A,H
SUB 04H
MOV H,A
SHLD CBIAS ; SAVE IT

; --- DETERMINE IF LINK IS ALREADY INSTALLED ---
LXI D,CONOUT+13 ; POINT TO 20K HOME ADDRESS
DAD D ; ADJUST FOR CP/M SIZE
LXI D,SHOME ; GET ADDRESS OF NEW VALUE
MOV A,M ; GET LOW ADDRESS BYTE
CMP E ; IS THERE A MATCH?
JNZ OK ; NO, SO LINK IS NOT INSTALLED
INX H ; POINT TO HIGH ADDRESS BYTE
MOV A,M ; GET IT
CMP D ; IS THERE A MATCH?
JNZ OK ; NO, SO LINK IS NOT INSTALLED
FILE: CLINK2 ASM PAGE 003

; LXI D,LMSG ; POINT TO ERROR MESSAGE
; MVI C,LST ; SET FOR LIST FUNCTION
; JMP BDOS ; LIST AND EXIT BACK TO CP/M

; --- COPY CODE UP TO 'FREE' LOCATION ----
OK:
LXI H,START ; SOURCE OF CODE
LXI D,FREE ; DESTINATION OF CODE
LXI B,LEN+2 ; LENGTH OF CODE
CALL MOVE

; --- COPY PART OF OLD BIOS JUMP TABLE UP TO LINK PGM ---
; LHLJ CBIA$ ; GET OFFSET
LXI D,4A16H ; 20K ADDRESS OF PART OF BIOS TABLE
DAD D ; ADJUST FOR CURRENT CP/M SIZE
PUSH H ; SAVE IT
LXI D,FHOM ; DESTINATION
LXI B,27
CALL MOVE
POP D

; --- COPY NEW LINK TABLE INTO BIOS JUMP TABLE ---
LXI H,NTAB ; NEW TABLE
LXI B,27 ; SET TABLE LENGTH
CALL MOVE

; --- PATCH IN LINK TO CONOUT ROUTINE ---
; LHLJ CBIA$ ; LOCATION OF CONOUT JUMP
DAD D ; CORRECT FOR CP/M SIZE
SHLD PTX+1 ; PATCH REFERENCES
SHLD PTX+1

; --- NOTIFY OF CORVUS LINK ---
; LXI D,BMSG
; MVI C,LST
; CALL BDOS

; ; DO A SYSTEM RESET
; MVI C,RSET
; JMP BDOS ; DO A RESET AND RE-ENTER CP/M (LINK IS DONE)

MOVE:
; MOV A,M
; STAX D
; INX H
; INX D
; DCX B
; MOV A,B
; ORA C
; JNZ MOVE
FILE: CLINK2 ASM PAGE 004

RET

; --- NEW JUMP TABLE TO BE COPIED INTO THE BIOS ---

NTAB:  JMP   SHOME ; JUMP TO SWITCH TABLE
        JMP   SELDSK
        JMP   SETTRK
        JMP   SETSEC
        JMP   SETDMA
        JMP   SREAD
        JMP   SWRITE
        JMP   FLISTST
        JMP   SSCTRAN

RMSG:  DB 0DH,0AH,' --- CORVUS LINK INSTALLED ---',ODH,0AH,'$'
LMSG:  DB 0DH,0AH,07,' ** CORVUS LINK ALREADY INSTALLED **',ODH,0AH,'$'
LDBIOS DS 2      ; BUFFER FOR BIOS LOCATION
CBIAS  DS 2      ; BUFFER FOR CP/M BIOS

START EQU $      ; START OF CODE TO BE MOVED UP

SHIFT EQU FREE-START ; OFFSET OF CODE TO MOVE UP LOCATION

NOTE: ALL LABELS IN THE CODE TO FOLLOW MUST BE
      OF THE FORM LABEL EQU $+SHIFT
      TO MAKE THE CODE CORRECTLY BE ASSEMBLED
      FOR THE SHIFTED ORIGIN (AT 'FREE').

; --- COPY OF ORIGINAL BIOS SELECT, SETTRK, SETSEC, AND SETDMA

FHOME EQU $+SHIFT
JMP   0
FSELECT EQU $+SHIFT
JMP   0 ; THIS GETS PATCHED ON STARTUP
FSTTRK EQU $+SHIFT
JMP   0
FSTSEC EQU $+SHIFT
JMP   0
FSTDMA EQU $+SHIFT
JMP   0
FREAD EQU $+SHIFT
JMP   0
FWRITE EQU $+SHIFT
JMP   0
FLISTST EQU $+SHIFT
JMP   0
FSCTRAN EQU $+SHIFT
JMP   0

; DS 2      ; EXTRA ROOM

; --- THIS JUMP TABLE IS USED AS A SWITCH TO DIRECT THE BIOS
; DISC INTERFACE CALLS TO THE FLOPPY OR HARD DISC ROUTINES.

SHOME EQU $+SHIFT
JMP   FHOME ; SET TO FLOPPY ROUTINES AT FIRST
SREAD EQU $+SHIFT
FILE: CLINK2 ASM PAGE 005

JMP FREAD
SWRITE EQU $+SHIFT
JMP FWRITE
SSCTRAN EQU $+SHIFT
JMP FSCTRAN

DS 2 ; EXTRA ROOM

; --- THIS JUMP TABLE IS USED TO COPY INTO THE SWITCHING
; JUMP TABLE TO LINK TO THE FLOPPY DISC (WITH THE
; SELECT ROUTINE).
;
FTAB EQU $+SHIFT
JMP FHOME
JMP FREAD
JMP FWRITE
JMP FSCTRAN

; --- THIS JUMP TABLE IS USED TO COPY INTO THE SWITCHING
; JUMP TABLE TO LINK TO THE HARD DISC (WITH THE
; SELECT ROUTINE).
;
HTAB EQU $+SHIFT
JMP HHOME
JMP HREAD
JMP HWRITE
JMP HSCTRAN

DS 2 ; EXTRA ROOM

; --- DISC PARAMETER BLOCKS ---
; THE EXAMPLE HERE DIVIDES ONE 9.7MBYTE CORVUS DISC INTO
; TWO LARGE PSEUDO DRIVES (OF EQUAL SIZE)
; AND ALSO PROVIDES FOR THE INTERFACE OF TWO FLOPPY DISC DRIVES
; OF ARBITRARY SIZE AND TYPE (THEY COULD EVEN BE OTHER HARD DISCS).
;
; DRIVE: A & B EXISTING CP/M 2.X SYSTEM (FLOPPIES?)
; DRIVE: C & D CORVUS HARD DISC.

; NOTE: THE NUMBERS SHOWN IN DPBC (THE PARAMETER BLOCK)
; FOR THE PSEUDO DRIVE AND ITS ASSOCIATED ALLOCATION
; BUFFER SIZES ARE THE RESULT OF CHOOSING:
; 37860 SECTORS/PSEUDO DRIVE
; 60 SECTORS/TRACK
; I RESERVED TRACK FOR OPERATING SYSTEM
; 256 DIRECTORY ENTRIES
; 8*1024 BYTE BLOCKS
;
DPBASE EQU $+SHIFT
DPEO EQU $+SHIFT ; CORVUS PSEUDO DRIVE 1
DW 0,0
DW 0,0
DW DBIRBUF,DPBC ; DIRECTORY BUFFER, PARAM. BLOCK
DW CSVO,ALVO ; CHECK, ALLOC MAP
FILE: CLINK2 ASM PAGE 006

DPE2 EQU $+SHIFT
DW 0,0 ; CORVUS PSEUDO DRIVE 2
DW 0,0
DW D1RBUF,DPBC ; DIRECTORY BUFFER, PARAMETER BLOCK
DW CSV1,ALVI ; CHECK, ALLOC MAP

;

DPBC EQU $+SHIFT
DW 60 ; SECTORS/TRACK ON CORVUS PSEUDO DRIVE
DB 6 ; BLOCK SHIFT
DB 63 ; BLOCK MASK
DB 3 ; EXTENT MASK
DW 589 ; DISK SIZE-1
DW 255 ; DIRECTORY MAX
DB 128 ; ALLOCO
DB 0 ; ALLOC1
DW 0 ; CHECK SIZE
DW 1 ; OFFSET

;

; ---- CORVUS DISC OFFSET TABLE ----

OFISBAS EQU $+SHIFT
PDRVO EQU $+SHIFT
DW 12 ; STARTING DISC ADDRESS FOR DRIVE 0
DB 0 ; THIS IS THE UPPER BYTE OF THE 20 BIT DISC ADDRESS
DB 1 ; ACTUAL PHYSICAL DRIVE # (1-4)

;

PDRV1 EQU $+SHIFT
DW 37884 ; STARTING DISC ADDRESS FOR DRIVE 1
DB 0
DB 1 ; ACTUAL PHYSICAL DRIVE # (1-4)

;

; ---- SECTOR TRANSLATION ROUTINE ----

HSCTRAN EQU $+SHIFT
MOV A,D ; TEST IF TABLE TRANSLATION IS REQUESTED
ORA E
JNZ STRI ; YES, SO DO IT
MOV L,C ; NO, SO JUST TRANSFER TO (H,L)
MOV H,B
RET

STRI EQU $+SHIFT
XCHG ; GET TABLE ADDRESS IN (H,L)
DAD B ; INDEX INTO TABLE
MOV L,M ; GET BYTE IN (H,L)
MVI H,0
RET

;

; ---- MESSAGE PRINTOUT ROUTINE ----

PTMSG EQU $+SHIFT
MOV A,H ; GET MESSAGE BYTE
CPI $ ; IS IT THE TERMINAL CHARACTER
RZ ; YES, SO RETURN
MOV C,A ; SAVE FOR CONSOLE OUTPUT
PUSH H
PTXD  EQU  $+SHIFT ; SETUP LOCATION FOR PATCH
CALL  CONOUT
POP   H
INX   H
JMP   FTMSG

; ---- CORVUS DISC READ ROUTINE ----

HREAD  EQU  $+SHIFT
MVI   A, RDCOM ; GET READ COMMAND
CALL  SETUP ; COMPUTE DISC ADDRESS AND ISSUE COMMANDS
LHLD  DMAAD ; GET DMA ADDRESS
RDCl  EQU  $+SHIFT
CALL  TURN ; WAIT FOR ACCEPTANCE OF COMMAND
JNZ   ERRCD ; IF ERROR
MVI   B, SSIZE ; GET SECTOR SIZE
RLP   EQU  $+SHIFT
IN    STAT ; GET DRIVE STATUS
ANI   DRDY ; LOOK AT READY BIT
JNZ   RLP ; LOOP UNTIL BYTE IS AVAILABLE
IN    DATA ; READ BYTE FROM CONTROLLER
MOV   M, A ; SAVE IT IN MEMORY
INX   H
DCR   B ; COUNT DOWN BYTES
JNZ   RLP ; LOOP UNTIL DONE
RTN   EQU  $+SHIFT
XRA   A ; CLEAR ERROR INDICATOR
RET

; ---- CORVUS DISC WRITE ROUTINE ----

HWRITE EQU  $+SHIFT
MVI   A, WRCOM ; GET WRITE COMMAND
CALL  SETUP ; COMPUTE ADDRESS AND ISSUE COMMANDS
MVI   B, SSIZE ; GET SECTOR SIZE
LHLD  DMAAD ; GET DMA ADDRESS
WLP   EQU  $+SHIFT
IN    STAT ; GET DRIVE STATUS
ANI   DRDY ; LOOK AT READY BIT
JNZ   WLP ; LOOP UNTIL BYTE ISS AVAILABLE
MOV   A, M ; GET BYTE FROM MEMORY
OUT   DATA ; SEND IT TO CONTROLLER
INX   H
DCR   B ; COUNT DOWN # OF BYTES
JNZ   WLP ; LOOP UNTIL DONE
CALL  TURN ; WAIT FOR BUS TURN AROUND AND READ ERROR #
JZ    RTN ; RETURN IF OK
ERRCD EQU  $+SHIFT
PUSH  B ; IF ERROR, ISSUE ERROR MESSAGE
LXI   H, EMSG
CALL  FTMSG
POP   PSW ; GET ERROR # BACK IN ACC
CALL  HEXIT ; PRINT IT OUT IN HEX
LXI   H, EMSG1
CALL  FTMSG ; PRINT REMAINDER OF MESSAGE
MVI   A, L ; SET ERROR INDICATOR
RET
TURN EQU $+SHIFT
IN STAT ; READ STATUS BYTE
ANI DIFAC ; LOOK AT DRIVE ACTIVE BIT
JNZ TURN
CALL DELAY1 ; WAIT FOR OVER 20USEC
CALL WAITI ; READ ERROR BYTE
MOV B,A ; SAVE IT
ANI $0H ; LOOK AT FATAL ERROR BIT
RET ;

DELAY1 EQU $+SHIFT
MVI B,6 ; DELAY MORE THAN 20USEC
DELAY EQU $+SHIFT
DCR B ; COUNT DOWN
JNZ DELAY
RET ;

WAITI EQU $+SHIFT
IN STAT ; GET STATUS BYTE
ANI DRDY ; LOOK AT READY BIT
JNZ WAITI
IN DATA ; GET DATA FROM CONTROLLER
RET ;

WAITO EQU $+SHIFT
MOV B,A ; SAVE COMMAND
IN STAT ; READ STATUS BYTE
ANI DRDY ; LOOK AT READY BIT
JNZ WAITO+1
MOV A,B ; GET COMMAND
OUT DATA ; SEND IT TO CONTROLLER
RET ;

; --- OUTPUT ACC IN HEX ---

HEXOT EQU $+SHIFT
PUSH PSW ; SAVE BYTE
RRC ; SHIFT UPPER NIBBLE DOWN 4 BITS
RRC
RRC
CALL HEXB ; OUTPUT UPPER NIBBLE IN HEX
POP PSW ; RESTORE BYTE
HEXB EQU $+SHIFT
ANI $0FH ; MASK OUT UPPER NIBBLE
ADI '0' ; ADD ASCII BIAS
CPI '9'+1 ; IS IT NUMERIC?
JC PRT ; YES, SO SEND IT OUT
ADI 7 ; NO, SO ADJUST FOR A-F
PRT EQU $+SHIFT
MOV C,A ; SAVE FOR OUTPUT
PTXI EQU $+SHIFT ; SETUP LOCATION FOR PATCH
JMP CONOUT ; OUTPUT TO CONSOLE

; --- COMPUTE CORSUS DISC ADDRESS AND SEND TO CONTROLLER ---


SETUP EQU $+SHIFT
CALL WAITO ; ISSUE DISC R/W COMMAND
LHLD TRACK ; GET TRACK # FROM BUFFER
XCHG ; PUT IN (D,E)
LXI H,0 ; CLEAR CONVERSION BUFFER
LDA NSPTRK ; GET # SECTORS/TRACK (ASSUMED <= 255)
MVI B,8 ; SET TO MULTIPLY 8 BITS
; MULTIPLY : (H.L)=TRACK* (# SECTORS/TRACK)
MULT EQU $+SHIFT
DAD H ; SHIFT BUFFER OVER 1 POSITION
RAL ; TEST NEXT BIT OF (#SECTORS/TRACK)
JNC ML1 ; IF NOT A 1, DON'T ADD IN
DAD D ; IF A 1, ADD IN TRACK #
ML1 EQU $+SHIFT
DCR B ; COUNT DOWN # BITS
JNZ MULT ; LOOP UNTIL DONE
XCHG ; PUT RESULT IN (D,E)
LHLD SECTOR ; GET SECTOR #
DAD D ; (H.L)=SECTOR+TRACK* (#SECTORS/TRACK)
; XCHG ; PUT RESULT IN (D,E)
LHLD ADDOF ; GET POINTER TO DISC ADDRESS OFFSET
; ADD IN DISC ADDRESS OFFSET
MOV A,E ; GET LOWER BYTE OF RELATIVE DISC ADDRESS
ADD M ; ADD IN LOWER BYTE OF ABSOLUTE DISC OFFSET
MOV E,A ; SAVE RESULT
INX H ; POINT TO NEXT BYTE OF OFFSET
MOV A,D ; DO ADDITION AGAIN
ADC M
MOV D,A ; SAVE IT
INX H ; POINT TO LAST BYTE OF OFFSET
MVI A,0 ; CLEAR ACC WITHOUT CLEARING CARRY BIT
ADC M ; GET UPPER BYTE OF DISC ADDRESS
RLC ; SHIFT OVER 4 PLACES
RLC
RLC
MOV C,A ; SAVE IT
LDA CODRIVE ; GET CORVUS DRIVE # (1-4)
ADD C ; MERGE IN EXTENDED DISC ADDRESS BITS
; WE NOW HAVE (D,E)=LOWER TWO BYTES OF DISC ADDRESS
; ACC = EXTENDED DISC ADDRESS+DRIVE #
; SET1 EQU $+SHIFT
CALL WAITO ; SEND DRIVE # TO CONTROLLER
MOV A,E
CALL WAITO ; SEND LOWER DISC ADDRESS TO CONTROLLER
MOV A,D
JMP WAITO
; --- HOME CORVUS DRIVE ----
; HHOME EQU $+SHIFT
LXI H,0 ; GET TRACK 0
SHLD TRACK
RET
; ---- SELECT DISC ROUTINE ----
; NOTE, THIS ROUTINE DOES A LOT OF EXTRA WORK SO
; THAT SOME OF IT NEED NOT BE DONE FOR EACH DISC
; READ/WRITE OPERATION. THE METHOD USED TO SWITCH
; BETWEEN CORVUS AND FLOPPY DRIVES (PATCHING A JUMP
; TABLE) IS MAINLY USED BECAUSE IT CONCENTRATES THE
; SELECT FUNCTIONS ALL WITHIN THE SELDSK ROUTINE.

SELDEN EQU $+SHIFT
MOV A,C  ; GET CP/M DRIVE #
CPI DMAX ; NO, SO SEE IF # IS TOO BIG
JNC SDLERR ; ERROR, SO GIVE NOTICE
CPI FMAX ; IS IT A FLOPPY?
JC SDL1 ; YES, SO PROCESS SELECT

COPY HARD DISC LINKS INTO SWITCH TABLE

LXI H,HTAB ; POINT TO HARD DISC TABLE
CALL COPYS ; DO IT

MOV A,C
ADD FMAX ; REMOVE FLOPPY OFFSET
MOV C,A
MOV L,C  ; GET CP/M DRIVE # IN (H,L)
MVI H,0
DAD H  ; MULTIPLY BY 4
DAD H
LXI D,OFDBAS ; POINT TO BASE OF OFFSET TABLE
DAD D  ; SELECT THE RIGHT ONE
SLFD ADDOFF ; SAVE POINTER FOR LATER USE
INX H
INX H
INX H
MOV A,M  ; GET ACTUAL CORVUS DRIVE #
STA CURDEV ; SAVE IT

SLD2 EQU $+SHIFT
MOV L,C  ; GET CP/M DRIVE # IN (H,L)
MVI H,0
DAD H  ; MULTIPLY BY 16
DAD H
DAD H
DAD H
LXI D,DPBASE ; GET START OF PARAM. BLOCK
DAD D  ; SELECT THE RIGHT BLOCK
SLFD PPOINT ; SAVE POINTER
LXI D,10
DAD D  ; POINT TO ADDRESS OF DISC BLOCK
MOV E,M  ; GET ADDRESS IN FROM TABLE INTO (D,E)
INX H
MOV D,M
XCHG ; PUT IN (H,L)
MOV E,M  ; GET # SECTORS/TRACK INTO (D,E)
INX H
MOV D,M
XCHG
SLFD NSPTRK ; SAVE IT IN BUFFER

SLD3 EQU $+SHIFT
LHLD PPOINT ; GET PARAM. POINTER
RET

; COPY FLOPPY JUMP TABLE INTO SWITCH TABLE
SLD EQU $+SHIFT
LXI H,FTAB ; POINT TO FLOPPY JUMP TABLE
CALL COPYS ; DO COPY
JMP FSELEC ; FINISH THRU FLOPPY SELECT ROUTINE

; SLDERR EQU $+SHIFT
LXI H,0 ; IF SELECT ERROR, GET 0 IN (H,L)
XRA A
STA CDISC ; SET TO REBOOT ON DRIVE A
RET

; COPY EQU $+SHIFT
LXI D,SHOME ; SET DESTINATION OF COPY (SWITCH TABLE)
MWI H,12 ; SET SIZE OF TABLE
COPY EQU $+SHIFT
MOV A,M ; GET BYTE FROM SOURCE
STAX D ; SAVE AT DESTINATION
INX H
INX D
DCR B
JNZ COPY
RET

; SETTRK EQU $+SHIFT
MOV L,C ; SAVE TRACK #
MOV H,B
SHLD TRACK
JMP FSTTRK ; DO FLOPPY ONE ALSO

; SETSEC EQU $+SHIFT
MOV L,C ; SAVE CP/M SECTOR #
MOV H,B
SHLD SECTOR
JMP FSTSEC ; DO FLOPPY ONE ALSO

; SETDMA EQU $+SHIFT
MOV L,C ; SAVE DMA ADDRESS
MOV H,B
SHLD DMAAD
JMP FSTDMA ; DO FLOPPY ONE ALSO

; ----------- MESSAGES -----------
;
ERMSG EQU $+SHIFT
DB $DF0H,0AH,07,' -- DISC R/W ERROR # $'
;
ERMSGI EQU $+SHIFT
DB 'n-',DF0H,0AH,'$'
;
; ----------- BUFFERS -----------
;
DMAAD EQU $+SHIFT
DS 2 ; DMA ADDRESS
TRACK EQU $+SHIFT
FILE: CLINK2  ASM  PAGE 012

DS  2  ; TRACK 
SECTOR EQU $+SHIFT
DS  2  ; SECTOR 
DSKNO EQU $+SHIFT
DB OFFH  ; CURRENT DISC # (UNDEFINED AT START)
ADDOF EQU $+SHIFT
DS  2  ; BUFFER FOR POINTER TO ADDRESS OFFSET
NSPTRK EQU $+SHIFT
DS  2  ; BUFFER WITH # SECTORS/TRACK
PPOINT EQU $+SHIFT
DS  2  ; POINTER TO CURRENT PARAM. BLOCK
CDrive EQU $+SHIFT
DS  1  ; BUFFER FOR CORVUS DISC 

DIRBUF EQU $+SHIFT
DS  128  ; DIRECTORY ACCESS BUFFER
ALVO EQU $+SHIFT
DS  74  ; DRIVE 0 ALLOC. MAP
CSV0 EQU $+SHIFT
DS  0  ; DRIVE 0 CHECK BUFFER (NOT USED)
ALVI EQU $+SHIFT
DS  74  ; DRIVE 1 ALLOC. MAP
CSV1 EQU $+SHIFT
DS  0  ; DRIVE 1 CHECK BUFFER

ENDP EQU $
LENC EQU ENDP-START  ; LENGTH OF CODE TO COPY
END
FILE: CDIAGNOS ASM  PAGE 001

; ----- CORVUS DISC DIAGNOSTIC PROGRAM ----- 
; VERSION 1.1
; BY BRK
;
; THIS PROGRAM PROVIDES A FEW RELATIVELY SAFE DISC DIAGNOSTICS 
; FOR THE CORVUS DRIVE. IT CONTAINS ITS OWN INSTRUCTIONS.
; FUNCTIONS AVAILABLE:
;
; 1. DISK FORMAT CHECK AND CORRECT (RESET CRC).
; 2. READ CONTROLLER CODE VERSION #.
; 3. HEAD SERVO TEST (FAST HEAD SEeks ACROSS DISC).
;
; NOTE: THE DISK FORMAT CHECK WILL ONLY WORK ON SYSTEMS WITH 
; CONTROLLER CODE VERSION # > 0. IF FOR SOME REASON YOU
; DO NOT WANT TO UPDATE IT FROM VERSION 0, BUT NEED TO
; FIX SOME BAD DISC SECTORS, YOU CAN USE CCODEX.COM TO 
; TEMPORARILY SWITCH CONTROLLER CODES TO RUN THIS PROGRAM 
; (FROM YOUR FLOPPY CP/M) THEN SWITCH BACK TO THE VERS. 0 
; CONTROLLER CODE.
;
;
; ----- CP/M EQUATES ----- 
;
BDOS EQU 05           ; BDOS ENTRY POINT
CHIN EQU 1            ; BDOS COMMAND FOR CONSOLE INPUT
CHOUT EQU 2           ; BDOS COMMAND FOR CONSOLE OUTPUT
LST EQU 9             ; BDOS COMMAND FOR WRITE LIST
;
CR EQU ODH            ; CARRIAGE RETURN
LF EQU 0AH             ; LINE FEED
;
;
; ----- CORVUS DISC EQUATES ----- 
;
DATA EQU ODEH         ; DATA I/O PORT
STAT EQU DATA+1       ; STATUS INPUT PORT
BRDY EQU 1            ; MASK FOR DRIVE READY BIT
DIFAC EQU 2           ; MASK FOR DRIVE ACTIVE BIT
VERCOM EQU 0          ; READ VERSION # AND # OF DRIVES COMMAND
FCKCOM EQU 7          ; FORMAT CHECK COMMAND
;
; DO NOT CHANGE RDCOM AND WRCOM WITHOUT ALSO CHANGING THE TEST IN 
; THE INIT ROUTINE.
;
RDCOM EQU 12H          ; READ COMMAND (FOR 128 BYTES/SECTOR)
WRCOM EQU 11H          ; WRITE COMMAND (FOR 128 BYTES/SECTOR)
COMOPS EQU 10H         ; R/W COMMAND OFFSET FROM VERS. 0 CONTROLLER CODE 
Ssize EQU 12H          ; SECTOR SIZE
;
;
ORG 100H               ; STANDARD CP/M TPA ORIGIN

START: LXI H,0          ; GET STACK POINTER IN (H,L)
DAD SP
SHLD $BUF  ; SAVE IT

; --- SETUP DIRECT CONSOLE I/O JUMPS ---
LHLD $1  ; GET ADDRESS OF WARM BOOT (BIOS+3)
LXI D, 3
DAD D  ; COMPUTE ADDRESS OF CONST
SHLD CONST+1  ; PATCH IN JUMP
DAD D
SHLD CONIN+1
DAD D
SHLD CONOUT+1
JMP SIGNON  ; SIGN ON AND START PROGRAM

; CONST: JMP 0  ; JUMP TO BIOS Routines
CONIN: JMP 0
CONOUT: JMP 0

; SIGNON: LXI SP, STACK  ; SETUP LOCAL STACK
LXI D, SMSC  ; POINT TO MESSAGE
CALL PTMSG  ; PRINT SIGN ON MESSAGE
Q1: LXI D, MSG2
CALL PTMSG  ; LIST TASK MENU
MNI: LXI D, MSG3
CALL PTMSG  ; ASK FOR CHOICE
CALL GTTSK  ; GET THE TASK
CPI 'O'
JZ Q1  ; IF LIST
CPI '1'
JZ INST  ; IF LIST INSTRUCTIONS
CPI '2'
JZ FCHK  ; IF FORMAT CHECK
CPI '3'
JZ RDCODE  ; IF READ VERSION #
CPI '4'
JZ SVRTST  ; IF SERVO TEST
JMP EXIT  ; EXIT BACK TO CP/M

; --- LIST INSTRUCTIONS COMMAND ---

; INST: LXI D, MSGI
JMP MNO

; --- READ CONTROLLER CODE COMMAND ---

; RDCODE: CALL INIT  ; INITIALIZE CONTROLLER AND READ VERSION #
RRC
RRC
RRC
RRC
PUSH PSW  ; SAVE IT
LXI D, MSGI
CALL PTMSG
POP PSW
CALL DECBT  ; OUTPUT IN DECIMAL
LXI D, CRLF
JMP MNO  ; BACK TO MENU

; --- DISC FORMAT CHECK COMMAND ---
FILE: CDIAGNOS ASM PAGE 003

; FCHK: LI X H, MSG10 ; POINT TO MESSAGE
         SHLD  MSGPTR ; SAVE IT
         CALL INIT ; INITIALIZE CONTROLLER AND READ VERSION #
         JNZ FCI ; IF NOT REV. 0, CONTINUE
         LXI D, MSG6
         CALL PTMSG
         JMP NNI

; FC1: CALL CTDREV ; ASK FOR AND GET DRIVE #
         LXI D, MSG10
         CALL PTMSG

; VERF: MV1 A, FCKCOM ; GET DISC FORMAT CHECK COMMAND
         CALL WAIT0 ; SEND IT
         LDA DRIVE ; GET DRIVE #
         CALL WAIT0 ; SEND IT

; VER1: IN STAT ; LOOK AT BUSG ACTIV BIT
         ANI BIFAC
         JZ TRN2 ; IF COMMAND IS FINISHED
         CALL KTST ; TEST FOR "CONFIDENCE MESSAGE"
         JMP VER1 ; LOOP UNTIL OK

; TRN2: MV1 B, 6 ; SET DELAY
         CALL DELAY
         CALL WERR1 ; TEST ERROR RETURN CODE
         JC MNI ; IF ERROR, RESTART
         CALL WAIT1 ; GET # OF DATA BYTES TO FOLLOW
         ORA A ; TEST IF NO ERRORS
         LXI D, MSG8
         JZ MNO ; ISSUE MESSAGE AND RESTART
         CPI 255 ; TEST IF TOO MANY BYTES
         LXI D, MSG9
         JZ MNO ; IF TOO MANY, ISSUE MSG AND RESTART
         MOV C, A ; SAVE COUNT
         RRC
         RRC
         ANI 3FH ; SAVE # OF ERRORS
         STA CTR
         LXI H, BUF ; POINT TO BUFFER
         CALL WAIT1 ; GET THE RETURN CODE
         MOV M, A ; SAVE ERROR BYTE
         INX H
         DCR C ; COUNT DOWN
         JNZ VER2 ; LOOP UNTIL DONE

; VER2: CALL WAIT1 ; GET THE RETURN CODE
         MOY M, A ; SAVE ERROR BYTE
         INX H
         DCR C ; COUNT DOWN
         JNZ VER2 ; LOOP UNTIL DONE

; VER3: LXI D, MSG7
         CALL PTMSG
         CALL ERRLST
         LXI D, CRLF
         JMP MNO

; --- HEAD SERVO TEST ---

; SVRTST: LXI H, MSGA ; POINT TO MESSAGE FOR KTST
         SHLD MSGPTR
         MV1 A, RDCOM
         STA RWCOM ; SET FOR READ MODE
         CALL CTDREV ; ASK FOR AND GET DRIVE #
CALL INIT ; INITIALIZE CONTROLLER AND FIX READ COMMAND
SVR1: LDA DRIVE ; GET DRIVE #
ANI OFH ; MASK OFF UPPER DISC ADDRESS
STA DRIVE
LXI H,0
SHLD DADD ; SET FOR DISC ADDRESS 0
LXI H,BUF ; POINT TO READ BUFFER
CALL RWSEC ; READ ONE SECTOR
;
CALL KTST
CPI 'C'-40H
JZ MNI ; TO STOP TEST
;
SVR2: LDA DRIVE ; GET DRIVE #
ADI 10H ; ADD IN UPPER DISC ADDRESS NIBBLE
STA DRIVE
LXI H,40204 ; LOWER PART OF DISC ADDRESS
SHLD DADD ; SET FOR DISC ADDRESS
LXI H,BUF ; POINT TO READ BUFFER
CALL RWSEC ; READ ONE SECTOR
JMP SVR1

-------- SUBROUTINES & DATA --------

--- VERIFY COMMAND ERROR LISTER ---

ERRLIST:
LXI H,BUF ; POINT TO START OF BUFFER
SHLD BFPTR ; SET BUFFER POINTER
ERRLIST1: MVI A,2 ; SET FOR 2 SPACES
CALL NSPACE ; PRINT (A) SPACES
CALL GTCHR ; GET CHAR FROM BUFFER
CALL DECHT ; PRINT IT OUT
MVI A,5 ; SET FOR 5 SPACES
CALL NSPACE
CALL GTCHR ; GET LOW BYTE OF CYLINDER #
MOV L,A
CALL GTCHR ; GET UPPER BYTE OF CYLINDER #
MOV H,A
CALL DECOI ; PRINT IT OUT IN DECIMAL
MVI A,5 ; SET FOR 5 SPACES
CALL NSPACE
CALL GTCHR ; GET TRACK SECTOR #
CALL DECHT ; OUTPUT IN DECIMAL
LXI D,CRLF
CALL PTMSG ; ISSUE CR/LF
LXI H,CTR ; POINT TO COUNTER
DCR M
JNZ ERRLIST1 ; LOOP UNTIL DONE
RET

GTCHR: PUSH H
LHLD BF PTR ; GET BUFFER POINTER
MOV A,H ; GET BYTE
INX     H ; INCREMENT POINTER
SHLD    BFPTR ; SAVE POINTER
POP     H
RET

; RST:
CALL    CONST ; TEST CONSOLE STATUS
ORA     A
RZ      ; RETURN IF NO KEY HAS BEEN HIT
CALL    CONIM ; OTHERWISE GET THE CHAR.
PUSH    PSW ; SAVE CHAR.
LHLD    MSGPTR ; GET POINTER TO MESSAGE
XCHG    
CALL    PTRSC ; PRINT IT OUT
POP     PSW ; GET CHAR. BACK
RET

; RWSEC:
LDA     RWCOM ; GET READ/ WRITE COMMAND
CALL    WAITO ; WAIT AND SEND IT
LDA     DRIVE ; GET DRIVE # AND HIGH ADD. NIBBLE
CALL    WAITO
LDA     DADD ; GET LOW BYTE OF DISC ADDRESS
CALL    WAITO
LDA     DADD+1 ; GET UPPER BYTE OF DISC ADDRESS
CALL    WAITO
LDA     RWCOM ; GET COMMAND AGAIN
CPI     WRCOM ; IS IT A WRITE COMMAND?
JZ      WRIT ; YES, SO WRITE A SECTOR
CALL    WERR ; NO, SO CHECK READ AND GET ERROR CODE
RC      ; RETURN IF ERROR

; RSEC:
LXI     B,SSIZE ; GET SECTOR SIZE
RLP:
IN      STAT ; READ STATUS PORT
ANI     DRDY
JNZ     RLP
IN      DATA ; READ BYTE FROM DISC
MOV     M,A ; SAVE IT IN MEMORY
INX     H
DCX     B
MOV     A,B
ORA     C
JNZ     RLP ; LOOP UNTIL DONE
RET

; WRIT:
LXI     B,SSIZE ; GET SECTOR SIZE
WLP:
IN      STAT ; READ STATUS PORT
ANI     DRDY
JNZ     WLP
MOV     A,M ; GET BYTE FROM MEMORY
OUT     DATA ; WRITE IT TO DISC
INX     H
DCX     B
MOV     A,B
ORA     C
JNZ     WLP ; LOOP UNTIL DONE

WERR:
CALL    TURN ; TURN AROUND BUSS

WERR1:
CALL    WATTI ; WAIT FOR ERROR BYTE
MOV B,A ; SAVE BYTE
ANI BOH ; LOOK FOR FATAL ERRORS
RZ ; OK, SO RETURN
PUSH B ; SAVE ERROR
LXI D,MSGE ; ERROR, SO ISSUE MESSAGE
CALL PFMSG
POP PSW ; GET ERROR BYTE BACK IN ACC
CALL HEXOT ; OUTPUT IN HEX
LXI B,MSGEI
CALL PFMSG

--- CANNOT AFFORD TO EXIT IF ERROR, SO TRY TO FIX IT ---

CALL INIT ; RE-SYNCHRONIZE CONTROLLER
STC ; SET CARRY TO INDICATE ERROR
RET

TURN: IN STAT
ANI DIFAC ; LOOK AT BUSS ACTIVE BIT
JNZ TURN
MVI B,6 ; GOOD AT 4MHZ ALSO
DELAY: DCR B
JNZ DELAY
RET

WAITI: IN STAT ; READ STATUS PORT
ANI DRDY ; LOOK AT READY LINE
JNZ WAITI ; LOOP UNTIL READY
IN DATA ; READ BYTE FROM DISC
RET

WAITO: PUSH PSW ; SAVE COMMAND
IN STAT ; READ STATUS PORT
ANI DRDY ; LOOK AT READY LINE
JNZ WAITO+1 ; LOOP UNTIL READY
POP PSW
OUT DATA ; WRITE BYTE TO DISC
RET

--- INITIALIZE CONTROLLER ----

INIT: MVI A,OFFH ; GET AN INVALID COMMAND
OUT DATA ; SEND IT TO CONTROLLER
MVI B,150 ; SET FOR LONG DELAY
CALL DELAY
IN STAT
ANI DIFAC ; LOOK AT DRIVE ACTIVE BIT
JNZ INIT ; LOOP UNTIL NOT ACTIVE
CALL WAITI ; GET ERROR CODE
CPI 8FH ; CHECK RETURN CODE
JNZ INIT ; IF NOT RIGHT, TRY AGAIN

TEST CONTROLLER CODE VERSION

MVI A,VERCOM ; GET COMMAND TO READ VERSION # AND # OF DRIVES
CALL WAITO ; SEND IT
CALL TURN ; WAIT FOR ACCEPTANCE
CALL W AiTi ; GET ANSWER
ANI OFOh ; MASK OUT 0 OF DRIVES
RMz
PUSH PSW ; SAVE IT AND FLAGS
LDA RWC0M ; GET READ/ WRITE COMMAND
SUI CMOfS ; SUBTRACT OFFSET TO REV. 0 CODE
STA RWC0M ; RESAVE IT
POP PSW
RET

--- MESSAGE PRINT ROUTINE---
PTMSG: MVI C, LIST ; CP/M WRITE LIST COMMAND
JMP BDOs ; EXECUTE BDOS COMMAND

--- OUTPUT BYTE IN ACC IN HEX ---
HEXOT: PUSH PSW ; SAVE BYTE
RRC ; SHIFT UPPER NIBBLE DOWN
RRC
RRC
CALL HEXB ; OUTPUT UPPER NIBBLE IN HEX
POP PSW ; GET BYTE BACK
HEXB: ANI OFH ; MASK OFF UPPER NIBBLE
ADI '0' ; ADD ASCII BIAS
CPI '9'+1 ; TEST IF NUMERIC
JC PRT ; YES, SO DO IT
ADI 7 ; NO, SO ADD BIAS FOR A-F
PRT: MOV C, A ; SETUP FOR OUTPUT
COUT: PUSH PSW
PUSH H ; BUFFERED CONSOLE OUTPUT
PUSH D
PUSH B
MOV E, C
MVI C, CHOUT ; BDOS CHAR. OUTPUT COMMAND
CALL BDOS
POP B
POP D
POP H
POP PSW
RET

--- OUTPUT (H,L) IN DECIMAL ---
DECOUT: LXI D, -10000 ; SET TO SUBTRACT 10000
MVI B, '0' ; SET TO SUPPRESS LEADING ZEROS
CALL DEC2 ; OUTPUT FIRST CHAR.
LXI D, -10000 ; SET TO SUBTRACT 1000
CALL DEC2 ; OUTPUT SECOND CHAR.
DEC4: LXI D, -100 ; SET TO SUBTRACT 100
CALL DEC2 ; OUTPUT THIRD CHAR.
LXI D, -10 ; SET TO SUBTRACT 10
CALL DEC2 ; OUTPUT FOURTH CHAR.
MVI B, 0 ; ALLOW LEADING ZERO
FILE: CDIAGNOS ASM

LXI D,-1 ; SET TO SUBTRACT 1
DEC2: MVI C,'0'-1 ; SET CHAR. COUNT
DEC3: SHLD DECBUF ; SAVE REMAINDER
INR C ; INC. ASCII CHAR. COUNTER
DAD D ; DO SUBTRACTION
JC DECO ; LOOP UNTIL UNDERFLOW
LHLD DECBUF ; GET LAST REMAINDER
MOV A,C ; GET CHAR. COUNTER
CMP B ; TEST FOR ZERO SUPPRESS
JZ SPACE ; ISSUE SPACE IF ZERO SUPPRESS IS ON
MVI B,'0' ; CLEAR ZERO SUPPRESS FLAG
JMP COUT ; OUTPUT CHAR.

; -- OUTPUT BYTE IN DECIMAL --

DECBT: PUSH H
PUSH D
PUSH B
MOV L,A ; SAVE BYTE IN (H,L)
MVI H,'0'
MVI B,'0' ; SET TO SUPPRESS LEADING ZEROS
CALL DECR
POP B
POP D
POP H
RET

; -- SPACE PRINTER FUNCTIONS --

SPACE: MVI A,1 ; SET FOR ONE SPACE
NSPACE: PUSH B
MOV B,A ; SAVE # OF SPACES TO OUTPUT
NSP1: MVI A,' ' ; GET A SPACE
CALL PRT ; PRINT IT OUT
DCR B ; COUNT DOWN
JNZ NSP1 ; LOOP UNTIL DONE
POP B
RET

; -- YES FUNCTION --

YES: CALL CONNC ; GET CONSOLE CHAR.
CPI 'Y' ; IS IT A Y?
JZ YES1
CPI 'N' ; IS IT A N?
JNZ YES ; IF NEITHER, KEEP TRYING
INR A ; SET N STATUS
YES1: PUSH PSW ; SAVE FLAGS
CALL CONOUT ; OUTPUT TO CONSOLE
POP PSW ; RESTORE FLAGS
RET

CONNC: CALL CONIN ; GET CHAR. FROM CONSOLE
MOV C,A ; SAVE FOR ECHO
CPI 60H ; IS IT LOWER CASE?
JC CON1 ; NO, SO CONTINUE
ANI 5FH ; YES, SO MASK TO UPPER CASE
CON1:
CPI 'C'-40H ; IS IT A CONTROL-C?
RMZ ; NO, SO RETURN

CTC: 
LXI D,CMGS ; POINT TO CONTROL-C MESSAGE

EXMG: 
CALL PMSG ; ISSUE MESSAGE

EXIT: 
LXI D,CRF
CALL PMSG ; ISSUE A CRF
LHLD SBUF ; GET OLD STACK POINTER
SPHL ; SET STACK
RET ; BACK TO CP/M

; -- GET COMMAND TASK --

GTTSK: 
CALL CONNC ; GET CONSOLE CHAR.
CPI '0'
JC GTTSK ; IF INVALID, TRY AGAIN
CPI '5'+1
JNC GTTSK

GTT1:
CALL COUT ; OK, SO ECHO
PUSH PSW ; SAVE IT
LXI D,CRF
CALL PMSG ; PRINT CRF
POP PSW
RET

; -- ASK FOR AND GET DRIVE # --

GTDRV: 
LXI D,DMPS
CALL PMSG ; ASK FOR DRIVE #

GTDRV1: CALL CONNC ; GET CONSOLE CHAR.
CPI '1'
JC GTDRV ; IF INVALID, TRY AGAIN
CPI '4'+1
JNC GTDRV1
CALL GTT1 ; ECHO AND CRF
SUI '0' ; REMOVE ASCII BIAS
STA DRIVE ; SAVE IT
RET

; --- MESSAGES ---

SMGO: 
DB CR,LF,' --- CORVUS DISC DIAGNOSTIC ---'
DB CR,LF,' ( VERSION 1.1 ) ','CR,LF,'$'

MSG2: 
DB CR,LF,' --- TEST MENU --- ','CR,LF
DB CR,LF,' 0. LIST THIS MENU'
DB CR,LF,' 1. LIST INSTRUCTIONS'
DB CR,LF,' 2. DISC FORMAT CHECK'
DB CR,LF,' 3. READ CONTROLLER CODE VERSION #'
DB CR,LF,' 4. HEAD SERVO TEST'
DB CR,LF,' 5. EXIT BACK TO CP/M (CTL-C ALSO WORKS) ','CR,LF,'$'

MSG3: 
DB CR,LF,' TASK (0 TO LIST) : $

MSG4: 
DB CR,LF,' HIT CONTROL-C TO STOP TEST ','CR,LF,'$
FILE: CDIAGNOS ASM PAGE 010

; MSG5: DB CR,LF,"TEST ABORTED",CR,LF,"$"
MSG6: DB CR,LF,07
DB '--- THIS FEATURE IS NOT AVAILABLE UNDER VERS. 0 CONTROLLER CODE'
CRLF: DB CR,LF,"$

; MSEG: DB CR,LF,CR,LF,07, '** DISC R/W ERROR # $'
MSEG1: DB 'H **',CR,LF,"$
DMSG: DB CR,LF," CORVUS DRIVE # (1-4) ? $
CMSC: DB 'C',CR,LF,"$

; MSG7: DB CR,LF,07, 'BAD SECTORS CORRECTED-',CR,LF
DB CR,LF," SURFACE CYLINDER SECTOR "
DB CR,LF, '# # #'
DB CR,LF,"----------------------'
DB CR,LF,"$

; MSG8: DB CR,LF," NO BAD SECTORS FOUND II",CR,LF,"$
; MSG9: DB CR,LF,07, 'OVER 63 BAD SECTORS FOUND AND RE-RITTEN ',CR,LF,"$
; MSG10: DB CR,LF," DISC FORMAT CHECK IN PROGRESS ',CR,LF,"$
; MSG11: DB CR,LF," CONTROLLER CODE VERSION $#-$'

; MSCI: DB CR,LF
DB CR,LF," THIS PROGRAM PROVIDES SOME RELATIVELY "SAFE" DISC"
DB CR,LF," DIAGNOSTICS FOR THE CORVUS DRIVE. THE FUNCTIONS"
DB CR,LF," AVAILABLE ARE: ',CR,LF
DB CR,LF," A. DISC FORMAT CHECK'
DB CR,LF," THE CONTROLLER TRYS TO READ EACH 512 BYTE"
DB CR,LF," SECTOR TO VERIFY THAT IT IS "GOOD" (HAS A"
DB CR,LF," CORRECT CRC). IF IT GETS A BAD CRC AFTER"
DB CR,LF," TWENTY READ ATTEMPTS, IT WILL RE-RITE THE'
DB CR,LF," SECTOR TO RESERT THE CRC. THIS USUALLY TAKES'
DB CR,LF," ABOUT ONE MINUTE. NOTE: THIS FUNCTION IS ONLY'
DB CR,LF," AVAILABLE ON SYSTEMS WITH CONTROLLER CODE VERSION'
DB CR,LF," NUMBER GREATER THAN ZERO.,CR,LF
DB CR,LF," B. READ THE CONTROLLER CODE VERSION $#-',CR,LF
DB CR,LF," HEAD SERVO TEST'
DB CR,LF," THIS TEST ALTERNATELY READS 128 BYTE SECTORS"'
DB CR,LF," AT DISC ADDRESS 0 AND 75740 UNTIL STOPPED"'
DB CR,LF," BY HITING A CONTROL-C.'
DB CR,LF,CR,LF,"$

; --- BUFFERS AND DATA ---

SRBUF: DS 2 ; OLD STACK POINTER
DAUD: DS 2 ; DISC ADDRESS
DRIVE: DS 1 ; DRIVE # AND ADDRESS NIBBLE
RWCOM: DS 4 ; READ/ WRITE COMMAND
CTR: DS 1 ; ERROR COUNTER
FILE: CDIAGNOS ASM  PAGE 011

BPFTR: DS 2 ; BUFFER POINTER
DBGBUF: DS 2 ; BUFFER FOR DECIMAL OUT ROUTINE
MSGPTR: DS 2 ; POINTER TO MESSAGE FOR KTST ROUTINE
DS 80 ; STACK SPACE
STACK EQU $0

; ORG (STACK+105H) AND OFF00H ; START ON PAGE BOUNDARY
BUF EQU $ ; BUFFER FOR 1 DISC SECTOR (128 BYTES)
;
END
FILE: CREFORM.ASM PAGE 001

; ----- CORVUS DISC SECTOR PERMUTATION PROGRAM -----;
; VERSION 1.1
; BY BRK
;
; THIS PROGRAM IS USED TO CONVERT DATA AND PROGRAMS STORED
; ON THE CORVUS DRIVE UNDER THE ORIGINAL S-100 CONTROLLER CODE
; TO A FORM COMPATIBLE WITH THE NEW CONTROLLER CODE RELEASED
; WITH "THE MIRROR". THE ORIGINAL CONTROLLER CODE (REV. 0, 9/79)
; AND THE NEW (REV. 1, 2/80) CODE REQUIRE A SLIGHTLY DIFFERENT
; ORDERING OF SECTORS ON THE DISC. THIS PROGRAM WILL PERMUTE
; THE DATA IN THE SECTORS TO THE NEW FORMAT. IT IS ONLY OF USE
; TO THOSE WHO HAVE DRIVES SHIPPED BEFORE 2/26/80 WHICH MAY
; HAVE THE OLD CONTROLLER CODE AND NEED TO BE UPDATED TO BE
; COMPATIBLE WITH "THE MIRROR" (OR OTHER NEW CORVUS PRODUCTS).
; THE PROGRAM CONTAINS ITS OWN DOCUMENTATION AND WILL ONLY RUN
; ON DRIVES WITH THE NEW CONTROLLER CODE (REV #>0). IF YOU HAVE
; A CORVUS DRIVE WITH THE REV. 0 CONTROLLER CODE (YOU CAN USE
; THE PROGRAM : CDIAGNS.COM TO READ THE CONTROLLER CODE #) AND
; WISH TO UPDATE IT, YOU MUST FIRST INSTALL THE NEW CONTROLLER
; CODE WITH THE PROGRAM: CCODE.COM.
;
; WARNING: ONCE THE SECTOR PERMUTATION IS STARTED IT MUST NOT
; BE STopped. IF YOU DO, YOU WILL HAVE A DISC THAT
; HAS DATA AND PROGRAMS THAT MAY BE PARTLY OF THE WRONG
; FORMAT FOR EITHER OF THE VERSIONS OF THE CONTROLLER
; CODE. IF THE PROGRAM IS ALLOWED TO RUN TO COMPLETION
; (ABOUT 34 MINUTES) ITS ACTIONS CAN BE REVERSED BY SIMPLY
; RUNNING THE PROGRAM AGAIN.
;
; ----- CP/M EQUATES -----;
BDOS EQU 05 ; BDOS ENTRY POINT
CHIN EQU 1 ; BDOS COMMAND FOR CONSOLE INPUT
CHOUT EQU 2 ; BDOS COMMAND FOR CONSOLE OUTPUT
LST EQU 9 ; BDOS COMMAND FOR WRITE LIST
;
CR EQU 0DH ; CARRIAGE RETURN
LF EQU 0AH ; LINE FEED
;
; ----- CORVUS DISC EQUATES -----
DATA EQU 0DEH ; DATA I/O PORT
STAT EQU DATA+1 ; STATUS INPUT PORT
DRDY EQU 1 ; MASK FOR DRIVE READY BIT
DDIFAC EQU 2 ; MASK FOR DRIVE ACTIVE BIT
VERCOM EQU 0 ; READ VERSION # AND # OF DRIVES COMMAND
FCKCOM EQU 7 ; FORMAT CHECK COMMAND
RDCOM EQU 32H ; READ COMMAND (FOR 512 BYTES/SECTOR)
WRCOM EQU 33H ; WRITE COMMAND (FOR 512 BYTES/SECTOR)
SSIZE EQU 512 ; SECTOR SIZE (USE THIS TO SPEED PROGRAM)
DSIZE EQU 18936 ; # OF 512 BYTES SECTORS ON THE DISC
FILE: CREFORM ASM PAGE 002

; ORG 100H ; STANDARD CP/M TPA ORIGIN

; START:
LXI   H,0
DAD   SP ; GET STACK POINTER IN (H,L)
SHLD  SBUF ; SAVE IT

; -- SETUP DIRECT CONSOLE I/O JUMPS ---
LHLD  I ; GET ADDRESS OF WARM ROOT (BIOS+3)
LXI   D,3
DAD   D ; COMPUTE ADDRESS OF CONST
SHLD  CONST+1 ; PATCH IN JUMP
DAD   D
SHLD  CONIN+1
DAD   D
SHLD  CONOUT+1
JMP   SIGNON ; SIGN ON AND START PROGRAM

; CONST:
CONIN: JMP 0 ; JUMP TO BIOS ROUTINES
CONOUT: JMP 0

; SIGNON:
LXI   SP,STACK ; SETUP LOCAL STACK
LXI   D,MSGS ; POINT TO MESSAGE
CALL  PTMSG ; PRINT SIGN ON MESSAGE
LXI   D,MSGC1
CALL  PTMSG ; PROMPT FOR INSTRUCTION
CALL  YES
JNZ   Q1 ; IF NO, CONTINUE
LXI   D,MSGC1
CALL  PTMSG ; LIST INSTRUCTIONS
Q1:  LXI   D,MSG2
CALL  PTMSG ; ASK IF OK TO DO IT
CALL  YES
JNZ   EXIT ; IF NOT, EXIT
LXI   D,MSG3
CALL  PTMSG ; ARE YOU SURE?
CALL  YES
JNZ   EXIT ; NO, SO EXIT
CALL  GTDRV ; ASK FOR AND GET DRIVE #

; LXI   D,CRLF
CALL  PTMSG ; ISSUE CRLF
CALL  INIT ; INITIALIZE CONTROLLER AND READ VERSION #
JNZ   VERF ; IF NOT REV. 0, CONTINUE
LXI   D,MSG5
JMP   EXNG ; EXIT WITH MESSAGE IF REV. 0 CODE

VERF:  MVI   A,FCKCOM ; GET DISC FORMAT CHECK COMMAND
CALL  WALT0 ; SEND IT
LDA   DRIVE ; GET DRIVE #
CALL  WALT0 ; SEND IT

VERFI: IN   STAT
ANI   DIFAC ; LOOK AT BUSS ACTIV BIT
JZ    TRN2 ; IF COMMAND IS FINISHED
CALL  KTST ; TEST FOR "CONFIDENCE MESSAGE"
JMP   VERFI ; LOOP UNTIL OK
TRN2: MVI B, 6 ; SET DELAY
CALL DELAY
CALL UNR-Disposition
JC VERF ; IF ERROR, TRY AGAIN
CALL WAITI ; GET # OF DATA BYTES TO FOLLOW
CSI 255 ; TEST IF TOO MANY BYTES
JZ VERF ; IF TOO MANY, TRY AGAIN
ORA A
JZ PERN ; IF NO BYTES EXPECTED, PROCEED
MOV C, A ; SAVE AS COUNTER
VER2: CALL WAITI ; GET THE RETURN CODES AND DISCARD THEM
DCR C ; COUNT DOWN
JNZ VER2 ; LOOP UNTIL DONE

PERM: LXI D, MSGA ; POINT TO "CONFIDENCE MESSAGE"
CALL PTMSG ; PRINT IT OUT
LXI H, DSIZE ; GET # OF 512 BYTE SECTORS ON DRIVE
SHLD WRLE ; SAVE IT
LXI H, 0
SHLD BADD ; SET STARTING DISK ADDRESS
CALL CONVT ; DO CONVERSION

EXMG: CALL PTMSG ; ISSUE MESSAGE
EXIT: LXI D, CRLF
CALL PTMSG ; ISSUE A CRLF
LHLD SBUF ; GET OLD STACK POINTER
SPHL ; SET STACK
RET ; BACK TO CP/M

----- SUBROUTINES & DATA ------

---- DO PERMUTATION OF SECTORS ----

CONVT: LXI H, BUFFER ; POINT TO BUFFER
MVI A, READCOM ; GET READ COMMAND
STA RWCOM ; SET R/W COMMAND
CALL RWSEC ; READ IN 1 SECTOR
JC CONVT ; TRY AGAIN IF ERROR

LXI H, BUFFER+128 ; POINT TO SECOND 128 BYTE SECTOR
LXI D, BUFFER+256 ; POINT TO THIRD 128 BYTE SECTOR
MVI C, 128 ; GET SECTOR SIZE
PLP: MOV B, H ; GET BYTE AND SAVE IT
LDAX D ; GET BYTE FROM THIRD SECTOR
MOV M, A ; PUT IT IN SECOND SECTOR
MOV A, B
STAX D ; COMPLETE PERMUTATION OF BYTES
INX H
INX D
DCR C
JNZ PLP ; COUNT DOWN SECTOR BYTES

WSEC: MVI A, WRCOM ; GET WRITE COMMAND
STA RWCOM ; SET TO WRITE
LXI    H, BUF
CALL   RWSEC
JC     WSEC
; POINT TO BUFFER
; WRITE SECTOR BACK TO DISC
; TRY AGAIN IF ERROR

LHLD   NBLKS
DCX    H
SHLD   NBLKS
MOV     A, H
ORA     L
RZ
; RETURN IF DONE
LHLD   DADD
INX    H
SHLD   DADD
; GET DISC ADDRESS
; UPDATE IT
CALL   KTST
; TEST IF "CONFIDENCE MESSAGE IS REQUESTED"

JMP    CONVT
; DO ANOTHER SECTOR

KTST:
CALL   CONST
ORA    A
RZ
; RETURN IF NO KEY HAS BEEN HIT
CALL   COMIN
; OTHERWISE GET THE CHAR.
LXI    D, MS4
; POINT TO "CONFIDENCE MESSAGE"
CALL   PTHSC
; PRINT IT OUT
RET

RWSEC:
LDA    RWCOM
CALL   WAITO
; GET READ/ WRITE COMMAND
CALL   WAITO
LDA    DADD
CALL   WAITO
LDA    DADD+1
CALL   WAITO
LDA    RWCOM
CPI    WRCOM
; IS IT A WRITE COMMAND?
JZ     WRIT
; YES, SO WRITE A SECTOR
CALL   WERR
; NO, SO ASSUME READ AND GET ERROR CODE
RSEC:
LXI    B, SSIZE
RLP:
IN     STAT
; READ STATUS PORT
ANI    DRDY
JNZ    RLP
IN     DATA
; READ BYTE FROM DISC
MOV     M, A
; SAVE IT IN MEMORY
INX    H
DCX    B
MOV     A, B
ORA     C
JNZ    RLP
; LOOP UNTIL DONE
RET

WRIT:
LXI    B, SSIZE
RLP:
IN     STAT
; READ STATUS PORT
ANI    DRDY
JNZ    RLP
FILE: CREFORM ASM PAGE 005

MOV A,M ; GET BYTE FROM MEMORY
OUT DATA ; WRITE IT TO DISC
INX H
DCX B
MOV A,B
ORA C
JNZ WLP ; LOOP UNTIL DONE
WERRE: CALL TURN ; TURN AROUND BUS
WERRI: CALL WAITI ; WAIT FOR ERROR BYTE
MOV B,A ; SAVE BYTE
ANI BH ; LOOK FOR FATAL ERRORS
RZ ; OK, SO RETURN
PUSH B ; SAVE ERROR
LXI D,MSGE ; ERROR, SO ISSUE MESSAGE
CALL PTHSG
POP PSW ; GET ERROR BYTE BACK IN ACC
CALL HEXOT ; OUTPUT IN HEX
LXI D,MSGEI
CALL PTHSG

; --- CANNOT AFFORD TO EXIT IF ERROR, SO TRY TO FIX IT ---
; CALL INIT ; RE-SYNCHRONIZE CONTROLLER
STC ; SET CARRY TO INDICATE ERROR
RET

TURN: IN STAT
ANI DIFAC ; LOOK AT BUS ACTIVE BIT
JNZ TURN
MVI B,6 ; GOOD AT 4MHZ ALSO
DELAY: DCR B
JNZ DELAY
RET

WAITI: IN STAT ; READ STATUS PORT
ANI DRRY ; LOOK AT READY LINE
JNZ WAITI ; LOOP UNTIL READY
IN DATA ; READ BYTE FROM DISC
RET

WAITO: PUSH PSW ; SAVE COMMAND
IN STAT ; READ STATUS PORT
ANI DRRY ; LOOK AT READY LINE
JNZ WAITO+1 ; LOOP UNTIL READY
POP PSW
OUT DATA ; WRITE BYTE TO DISC
RET

; --- INITIALIZE CONTROLLER ---
; INIT: MVI A,OFFH ; GET AN INVALID COMMAND
OUT DATA ; SEND IT TO CONTROLLER
MVI B,130 ; SET FOR LONG DELAY
CALL DELAY
IN STAT
ANI DIFAC ; LOOK AT DRIVE ACTIVE BIT
JNZ INIT ; LOOP UNTIL NOT ACTIVE
CALL WAITI ; GET ERROR CODE
CPI 8FH ; CHECK RETURN CODE
JNZ INIT ; IF NOT RIGHT, TRY AGAIN

; TEST CONTROLLER CODE VERSION

MVI A,VERCOM ; GET COMMAND TO READ VERSION # AND # OF DRIVES
CALL WAITO ; SEND IT
CALL TURN ; WAIT FOR ACCEPTANCE
CALL WAITI ; GET ANSWER
ANI OFH ; MASK OUT # OF DRIVES
RET

; --- MESSAGE PRINT ROUTINE---

PTMSG: MVI C,LST ; CP/M WRITE LIST COMMAND
JMP BDOS ; EXECUTE BDOS COMMAND

; --- OUTPUT BYTE IN ACC IN HEX ---

HEXOT: PUSH PSW ; SAVE BYTE
RRC RRC RRC RRC
CALL HEXB ; OUTPUT UPPER NIBBLE IN HEX
POP PSW ; GET BYTE BACK
HEXB: ANI OFH ; MASK OFF UPPER NIBBLE
ADI '0' ; ADD ASCII BIAS
CPI '9'+1 ; TEST IF NUMERIC
JC PRT ; YES, SO DO IT
ADI 7 ; NO, SO ADD BIAS FOR A-F
PRT: MOV C,A ; SETUP FOR OUTPUT
COUT: PUSH PSW
PUSH H ; BUFFERED CONSOLE OUTPUT
PUSH D
PUSH B
CALL CONOUT
POP B
POP D
POP H
POP PSW
RET

; --- YES FUNCTION ---

YES: CALL CONNC ; GET CONSOLE CHAR.
CPI 'Y' ; IS IT A Y?
JZ YESI
CPI 'N' ; IS IT A N?
JNZ YES ; IF NEITHER, KEEP TRYING
INR A ; SET N STATUS
YESI: PUSH PSW ; SAVE FLAGS
CALL CONOUT ; OUTPUT TO CONSOLE
FILE: CREFORM ASH  PAGE 007

POP    PSW    ; RESTORE FLAGS
RET

CONNC: CALL    CONIN    ; GET CHAR. FROM CONSOLE
MOV    C,A    ; SAVE IT
CPI    60H    ; IS IT LOWER CASE?
JC     CONI    ; NO, SO CONTINUE
ANI    5FH    ; YES, SO MASK TO UPPER CASE
CONI:   CPI    'C'-40H    ; IS IT A CONTROL-C?
RNZ    ; NO, SO RETURN

CTC:   LXI    D,CMSC    ; POINT TO CONTROL-C MESSAGE
JMP    EXHG    ; ISSUE IT AND EXIT

; --- ASK FOR AND GET DRIVE # ---

; GTDRV:    LXI    D,DMSC    ; ASK FOR DRIVE #
CALL    PTMSG

; GTDRV1:   CALL    CONNC    ; GET CONSOLE CHAR.
CPI    '1'
JC     GTDRV1    ; IF INVALID, TRY AGAIN
CPI    '4'+1
JNC    GTDRV1
SUI    '0'    ; REMOVE ASCII BIAS
STA    DRIVE    ; SAVE IT
CALL    COUT    ; ECHO IT
LXI    D,CRLF
JMP    PTMSG

; --- MESSAGES ---

; SMSG:    DB    CR,LF, ' '    -- CORVUS SECTOR FORMAT UPDATE PROGRAM ---
DB    CR,LF, ' '    ( VERSION 1.1 ) ' ,CR,LF,'$'

; MSG1:    DB    CR,LF, ' '    DO YOU WANT THE INSTRUCTIONS (Y/N) ? $'

; MSG2:    DB    CR,LF,CR,LF,' '    CONVERT DATA ON DISC (Y/N) ? $'

; MSG3:    DB    CR,LF,CR,LF,' '    ARE YOU SURE (Y/N) ? $'

; DMSC:    DB    CR,LF,CR,LF,' '    CORVUS DRIVE # (1-4) ? $'

; MSG4:    DB    CR,LF,' '    DISC SECTOR FORMAT CONVERSION IN PROGRESS ' ,CR,LF
CRLF:    DB    CR,LF,'$'

; MSG5:    DB    CR,LF,CR,LF,07
DB    ' '    THIS PROGRAM WILL NOT RUN UNDER REV. 0 CONTROLLER CODE ---
DB    CR,LF,'$'

; MSG6:    DB    CR,LF,7,' '    THE SECTOR FORMAT CONVERSION IS NOW DONE ' ,CR,LF,7,'$'

; MSGE:    DB    CR,LF,CR,LF,07,' '    ** DISC R/W ERROR # $'

; MSGE1:    DB    'H'    ' '    ,CR,LF,'$'

; CMSG:    DB    'C',CR,LF,'$'
MSG1: DB CR,LF
    ' THIS PROGRAM IS TO BE USED TO CHANGE THE ORDER'
DB CR,LF,' OF THE 128 BYTE SECTORS ON THE CORVUS DRIVE. THIS'
DB CR,LF,' IS NEEDED WHEN UPDATING A DRIVE WITH THE NEW'
DB CR,LF,' CONTROLLER CODE RELEASED WITH "THE MIRROR". THIS'
DB CR,LF,' NEW CODE ALLOWS FOR VARIABLE SECTOR SIZES (128, 256,'DB CR,LF,' AND 512 BYTE SECTORS) AND NEW COMMANDS FOR "THE MIRROR".'
DB CR,LF,' UNFORTUNATELY, THE ORIGINAL 128 BYTE/SECTOR FORMAT'
DB CR,LF,' (REV. 0 OF THE CONTROLLER CODE, SHIPPED ON DRIVES PRIOR'
DB CR,LF,' TO 2/26/80) IS NOT UPWARDS COMPATIBLE WITH THIS NEW'
DB CR,LF,' FORMAT. IF YOU WISH TO UPDATE THE CONTROLLER CODE ON'
DB CR,LF,' A CORVUS DRIVE WHICH CONTAINS DATA WRITTEN IN THE OLD'
DB CR,LF,' FORMAT, YOU HAVE THREE CHOICES: ', CR,LF
DB CR,LF,' 1. THROW AWAY THE THE OLD DATA AND PROGRAMS ON THE'
DB CR,LF,' DISC AND START FROM SCRATCH WITH THE NEW CONTROLLER
DB CR,LF,' CODE. (NOT USUALLY A GOOD SOLUTION)
DB CR,LF,' 2. COPY ALL OF THE DATA AND PROGRAMS ON THE HARD DISC'
DB CR,LF,' TO SOME EXTERNAL STORAGE MEDIUM (MAGTAPE, FLOPPY'
DB CR,LF,' DISC... 15.5 MILES OF PAPER TAPE), SWITCH THE
DB CR,LF,' CONTROLLER CODE, AND RESTORE THE DATA AND PROGRAMS.
DB CR,LF,' 3. SWITCH TO THE NEW CONTROLLER CODE AND USE THIS
DB CR,LF,' PROGRAM TO REFORMAT THE DATA ON THE DISC. ', CR,LF
DB CR,LF,' WE BELIEVE THAT THE LAST CHOICE IS THE SIMPLEST (AND'
DB CR,LF,' POSSIBLY THE BEST) SOLUTION PROVIDED THAT YOUR COMPUTER'
DB CR,LF,' SYSTEM IS RELIABLE. THIS PROGRAM READS IN ALL 75744
DB CR,LF,' 128 BYTE SECTORS (ACTUALLY 18936 IN THE 512 BYTE/SECTOR
DB CR,LF,' MODE) AND INTERCHANGES THE MIDDLE TWO SECTORS OUT OF EVERY
DB CR,LF,' FOUR. THUS, THE ACTION OF THE PROGRAM CAN BE REVERSED BY
DB CR,LF,' SIMPLY RUNNING IT A SECOND TIME (IF FOR SOME REASON YOU
DB CR,LF,' WANTED TO GO BACK TO THE OLDER FORMAT). ', CR,LF
DB CR,LF,' TO USE THE PROGRAM: ', CR,LF
DB CR,LF,' 1. USE THE PROGRAM: CCODE.COM TO INSTALL THE NEW
DB CR,LF,' CONTROLLER CODE.
DB CR,LF,' 2. RUN THIS PROGRAM. A CONTROL-C IN RESPONSE TO A'
DB CR,LF,' QUERY WILL CAUSE AN EXIT BACK TO CP/M. ONCE
DB CR,LF,' THE DISC OPERATIONS HAVE BEGUN, HITTING ANY KEY ON
DB CR,LF,' THE CONSOLE WILL RESULT IN A "CONFIDENCE MESSAGE"'
DB CR,LF,' PRINT OUT INDICATING THAT THE PROGRAM IS STILL
DB CR,LF,' WORKING. ', CR,LF
DB CR,LF,' NOTE: IT IS PROBABLY A GOOD IDEA TO BACK UP YOUR MOST'
DB CR,LF,' IMPORTANT FILES JUST IN CASE SOMETHING GOES WRONG.'
DB CR,LF,' IF YOUR SYSTEM IS RUNNING RELIABLY, THIS PROGRAM TAKES
DB CR,LF,' ABOUT 34 MINUTES TO RUN TO COMPLETION. IT MUST NOT
DB CR,LF,' BE INTERRUPTED BECAUSE THIS WOULD LEAVE PART OF THE DISC'
DB CR,LF,' WITH THE WRONG FORMAT. ', CR,LF,' $'

BUFFERS AND DATA ----

SBUF: DS 2 ; OLD STACK POINTER
DADD: DS 2 ; DISC ADDRESS
DRIVE: DS 1 ; BUFFER FOR DRIVE #
NBLSK: DS 2 ; # DISC SECTORS TO R/W
RWCOM: DS 1 ; READ/ WRITE COMMAND
DS 80 ; STACK SPACE
STACK EQU $
ORG (STACK+105H) AND OFF00H ; START ON PAGE BOUNDARY
BUF EQU $ ; BUFFER FOR 1 DISC SECTOR (512 BYTES)
END
FILE: CODE ASM PAGE 001

;----- CORVUS CONTROLLER CODE UPDATE PROGRAM -----
; VERSION 1.2
; BY BRK
;
; THIS PROGRAM IS USED TO UPDATE THE CONTROLLER CODE ON THE
; CORVUS DISC. IT READS IN THIS CODE FROM A DISC FILE (USUALLY
; ON A CP/M FLOPPY DISC), LISTS ITS ASCII HEADER, AND Optionally
; WRITES IT TO THE CORVUS DRIVE. IT CONTAINS ITS OWN INSTRUCTIONS.
;
; WARNING: DO NOT WRITE THE CODE OUT TO THE DISC WITHOUT ADDING
; THE JUMPER BETWEEN PINS: D37 & D38 ON THE BACKPLANE
; OF THE DRIVE. IF YOU DO, IT WILL WRITE THE CODE
; OUT TO THE USUAL USER AREA OF THE DISC - OVERLAYING
; USER PROGRAMS, DATA, AND DIRECTORY DATA (THE PROGRAM
; WILL PROMPT FOR PERMISSION BEFORE WRITING THE CODE
; TO THE DISC).
;
; ----- CP/M EQUATES -----
;
FCB EQU 5CH ; STD FCB
BDOS EQU 05 ; BDOS ENTRY POINT
OFST EQU 806H ; CCP OFFSET FROM BDOS ENTRY POINT
CHIN EQU 1 ; BDOS COMMAND FOR CONSOLE INPUT
CHOUT EQU 2 ; BDOS COMMAND FOR CONSOLE OUTPUT
OPEN EQU 15 ; BDOS COMMAND TO OPEN FILE FOR READING
SRCH EQU 17 ; BDOS COMMAND TO SEARCH FOR FILE
READ EQU 20 ; BDOS COMMAND TO READ A SECTOR
SDMA EQU 26 ; BDOS COMMAND TO SET DMA ADDRESS
;
CR EQU 0DH ; CARRIAGE RETURN
LF EQU 0AH ; LINE FEED
;
; ----- CORVUS DISC EQUATES -----
;
DATA EQU 0DEH ; DATA I/O PORT
STAT EQU DATA+1 ; STATUS INPUT PORT
DRDY EQU 1 ; MASK FOR DRIVE READY BIT
DIFAC EQU 2 ; MASK FOR DRIVE ACTIVE BIT
WRCOM EQU 3 ; CONTROLLER ROM WRITE CODE
DRIVE EQU 1 ; DRIVE # FOR WRITING TO
FSIZE EQU 512 ; SECTOR SIZE FOR CONTROLLER CODE WRITE
FSIZE EQU 23 ; NUMBER OF 512 BYTE SECTORS FOR CONT. CODE
;
ORG 100H ; STANDARD CP/M TPA ORIGIN
;
START: LXI H,0
DAD SP ; GET STACK POINTER IN (H,L)
SHLD SBUF ; SAVE IT
;
--- SETUP DIRECT CONSOLE I/O JUMPS ---
LHLD 1 ; GET ADDRESS OF WARM BOOT (BIOS+3)
LXI D,3
DAD D ; COMPUTE ADDRESS OF CONST
FILE: CCODE  ASM  PAGE 302

SHLD  CONST+1  ; PATCH IN JUMP
DAD  D
SHLD  CONIN+1
DAD  D
SHLD  CONOUT+1
JMP  SIGNON  ; SIGN ON AND START PROGRAM

; CONST:  JMP  0  ; JUMP TO BIOS Routines
CONIN:  JMP  0
CONOUT:  JMP  0

SIGNON: LXI  SP,STACK  ; SETUP LOCAL STACK
LXI  D,SHMSG  ; POINT TO MESSAGE
CALL  PTMSG  ; PRINT SIGN ON MESSAGE
LXI  D,MSG1
CALL  PTMSG  ; PROMPT FOR INSTRUCTION
CALL  YES
JNZ  TFREE  ; IF NO, TEST FILE NAME
LXI  D,MSG1
CALL  PTMSG  ; IF YES, POINT TO INSTRUCTIONS
PRINT THEM OUT
TFREE:  LDA  FCB+1  ; GET FIRST CHAR. OF FILE NAME
CPI  ' '  ; IS IT A SPACE?
JZ  NERR  ; YES, NO NAME GIVEN
JC  NERR  ; IF BAD NAME

; LXI  D,TYP  ; POINT TO DESIRED TYPE (.CLR)
LXI  H,FCB+9  ; POINT TO FILE TYPE
MVII  C,3  ; LENGTH OF FILE TYPE
CALL  COMPARE  ; TEST FILE TYPE
JNZ  NERR  ; IF ERROR

OPENF: LXI  D,FCB  ; POINT TO FCB
MVII  C,OPEN  ; GET OPEN COMMAND
CALL  BDOS  ; OPEN FILE
INR  A
JNZ  RDIT  ; IF PRESENT, READ IT IN
LXI  D,MSG7
CALL  PTMSG  ; ISSUE FILE NOT FOUND MSG
NERR:  LXI  D,MSG5  ; COMMAND FORMAT MESSAGE
JMP  EXMG  ; PRINT MESSAGE AND EXIT

; RDIT:  XRA  A
STA  FCB+32  ; INSURE THAT IT STARTS AT FIRST RCD.
STA  WPLG  ; CLEAR CCP OVERLAY FLAG
CALL  RDCODE  ; LOAD CODE INTO MEMORY BUFFER

; LHLR  RADD  ; GET LAST DMA LOCATION
LXI  D,130+OFST  ; GET OFFSETS
DAD  D
XCHG
LHLR  BDOS+1  ; GET LOCATION OF BDOS ENTRY
XCHG
MOV  A,L
SUB  E
MOV  A,H
SBB  D
JC  RDIT1  ; IF NO OVERLAY OF CCP, PROCEED
MVI    A,1
STA    WFLG ; IF OVERLAY, SET TO WARM BOOT

; REDUX:
LXI    H,BUFF; POINT TO START OF BUFFER
LXI    D,TEST ; POINT TO EXPECTED TEST CODE
MVI    C,9 ; LENGTH OF CODE
CALL    COMPARE ; COMPARE THEM
JNZ    EXNG ; ISSUE IT IF COMPARE ERROR

LXI    D,MSG4
CALL    PTMSG ; PRINT LABEL
LXI    D,BUFF ; PRINT IT OUT
CALL    PTMSG ; POINT TO START OF CODE
INX    D ; SAVE POINTER
XCHG
SHLD    CODE
SHLD    RADD
LXI    D,MSG31
CALL    PTMSG ; BRACKET HEADER MESSAGE

LXI    D,JMSG
CALL    PTMSG ; ASK IF JUMPER IS INSTALLED
CALL    YES
JNZ    EXIT ; EXIT IF NO JUMPER
LXI    D,MSG2
CALL    PTMSG ; WRITE CODE TO DISC?
CALL    YES
JZ    WTIT ; YES, DO IT
JMP    EXIT ; NO, SO EXIT

; WTIT:
LXI    H,0
SHLD    DADD ; SET DISC ADDRESS
LXI    H,C SIZE ; # OF 512 BYTE SECTORS
SHLD    NbLKS
CALL    WTCODE ; WRITE CODE TO CORVUS DRIVE
LXI    H,24
SHLD    DADD ; SET DISC ADDRESS
LXI    H,C SIZE
SHLD    NbLKS ; SET # OF BLOCKS
LHLD    CODE ; SET RAM ADDRESS
SHLD    RADD
CALL    WTCODE
LXI    D,MSG6 ; POINT TO EXIT MESSAGE
CALL    PTMSG ; PRINT MESSAGE

; EXMG:
LXI    D,CRLF
CALL    PTMSG ; ISSUE CRLF
LXI    D,80H ; DMA ADDRESS
MVI    C,SDMA
CALL    BDQS ; RESET DMA ADDRESS
LHLD    SBUF ; GET OLD STACK POINTER
SPHL
LDA    WFLG ; GET OVERLAY FLAG
ORA    A ; TEST IT
RZ    ; OK, SO BACK TO CP/M
JMP    O ; IF CP/M OVERLAY, WARM BOOT
FILE: CCODE    ASM    PAGE 004

; ----- SUBROUTINES & DATA -----

; --- WRITE A BLOCK OF CODE TO THE HARD DISC ---

WTODE:   LHL   RADD   ; GET RAM ADDRESS
         CALL   WTSEC   ; WRITE A SECTOR
         SHLD   RADD
         LHL   NBLKS
         DCX   H
         SHLD   NBLKS
         MOV   A,H
         ORA   L
         RZ    ; RETURN IF DONE
         LHL   DADD   ; GET DISC ADDRESS
         INX   H
         SHLD   DADD   ; UPDATE IT
         JMP   WTODE   ; DO ANOTHER SECTOR

WTSEC:   MV   A,WRCOM   ; GET WRITE COMMAND
         CALL   WAITO   ; WAIT AND SEND IT
         MV   A,DRIVE   ; GET DRIVE #
         CALL   WAITO
         LDA   DADD   ; GET LOW BYTE OF DISC ADDRESS
         CALL   WAITO
         LDA   DADD+1   ; GET UPPER BYTE OF DISC ADDRESS
         CALL   WAITO

WRIT:    LXI   B,SIZE   ; GET SECTOR SIZE
         IN   STAT   ; READ STATUS PORT
         ANI   RDY
         JNZ   WLP
         MOV   A,M   ; GET BYTE FROM MEMORY
         OUT   DATA   ; WRITE IT TO DISC
         INX   H
         DCX   B
         MOV   A,B
         ORA   C
         JNZ   WLP   ; LOOP UNTIL DONE
         CALL   TURN   ; TURN AROUND BUS
         CALL   WAITI   ; WAIT FOR ERROR BYTE
         MOV   B,A   ; SAVE BYTE
         ANI   BUH   ; LOOK FOR FATAL ERRORS
         RZ   ; OK, SO RETURN
         PUSH   B   ; SAVE ERROR
         LXI   D,MSGE   ; ERROR, SO ISSUE MESSAGE
         CALL   PTMSG
         POP   PSW   ; GET ERROR BYTE BACK IN ACC
         CALL   HEXOT   ; OUTPUT IN HEX
         LXI   D,MSGEI
         CALL   PTMSG
         JMP   EXIT

TURN:    IN   STAT
         ANI   DIFAC   ; LOOK AT BUSY ACTIVE BIT
         JNZ   TURN
FILE: CCODE    ASM    PAGE 005

MVI    B, 6    ; GOOD AT 4MHZ ALSO

DELAY:
  DCK    B
  JNZ    DELAY
  RET

; WAITI:
  IN    STAT    ; READ STATUS PORT
  ANI    DRDY    ; LOOK AT READY LINE
  JNZ    WAITI    ; LOOP UNTIL READY
  IN    DATA    ; READ BYTE FROM DISC
  RET

; WAITO:
  PUSH    PSW    ; SAVE COMMAND
  IN    STAT    ; READ STATUS PORT
  ANI    DRDY    ; LOOK AT READY LINE
  JNZ    WAITO+1    ; LOOP UNTIL READY
  POP    PSW
  OUT    DATA    ; WRITE BYTE TO DISC
  RET

; --- INITIALIZE CONTROLLER ----

INIT:
  MVI    A, OFFH    ; GET AN INVALID COMMAND
  OUT    DATA    ; SEND IT TO CONTROLLER
  MVI    B, 150    ; SET FOR LONG DELAY
  CALL    DELAY
  IN    STAT
  ANI    DIFAC    ; LOOK AT DRIVE ACTIVE BIT
  JNZ    INIT    ; LOOP UNTIL NOT ACTIVE
  CALL    WAITI    ; GET ERROR CODE
  CPI    BFH    ; CHECK RETURN CODE
  JNZ    INIT    ; IF NOT RIGHT, TRY AGAIN
  RET

; --- MESSAGE PRINT ROUTINE ---
; THIS IS USED INSTEAD OF USUAL FUNCTION CODE #9
; SO THAT THE POINTER TO END OF LIST CAN BE RECOVERED.

PTMSG:
  LDAX    D    ; GET CHARACTER
  CPI    ' $'    ; IS IT END CHAR. ?
  RZ    ; YES, EXIT
  PUSH    D    ; SAVE POINTER
  MOV    E, A    ; SAVE FOR OUTPUT
  MVI    C, CHOUT    ; CONSOLE OUTPUT CODE
  CALL    B DOS    ; OUTPUT CHAR. TO CONSOLE
  POP    D
  INX
  JMP    PTMSG    ; LOOP TO OUTPUT ALL OF LIST

; --- OUTPUT BYTE IN ACC IN HEX ---

HEXOT:
  PUSH    PSW    ; SAVE BYTE
  RRC    ; SHIFT UPPER NIBBLE DOWN
  RRC
  RRC
  RRC
  JMP

EOF
RRC
CALL HEXB ; OUTPUT UPPER NIBBLE IN HEX
POP PSW ; GET BYTE BACK
HEXB:
ANI 0FH ; MASK OFF UPPER NIBBLE
ADI '0' ; ADD ASCII BIAS
CPI '9'+1 ; TEST IF NUMERIC
JC PRT ; YES, SO DO IT
ADI 7 ; NO, SO ADD BIAS FOR A-F
PRT:
MOV C,A ; SETUP FOR OUTPUT
COUT:
PUSH PSW
PUSH H ; BUFFERED CONSOLE OUTPUT
PUSH D
PUSH B
CALL CONOUT
POP B
POP D
POP H
POP PSW
RET

; ; -- YES FUNCTION --
;
YES:
CALL CONNC ; GET CONSOLE CHAR.
CPI 'Y' ; IS IT A Y?
JZ YES1
CPI 'N' ; IS IT A N?
JNZ YES ; IF NEITHER, KEEP TRYING
INR A ; SET N STATUS
YES1:
PUSH PSW ; SAVE FLAGS
CALL CONOUT ; OUTPUT TO CONSOLE
POP PSW ; RESTORE FLAGS
RET
CONNC:
CALL CONIN ; GET CONSOLE CHAR.
MOV C,A ; SAVE FOR ECHO
CPI 60H ; IS IT LOWER CASE?
JC CON1 ; NO, SO CONTINUE
ANI 5FH ; YES, SO MASK TO UPPER CASE
CON1:
CPI 'C'-'40H ; IS IT A CONTROL-C?
RNZ ; NO, SO RETURN
CTC:
LXI D,CMNG ; POINT TO CONTROL-C MESSAGE
JMP EXMG ; ISSUE IT AND EXIT

; ; --- READ IN CODE FROM CP/M DISC ---
;
RDCODE:
LXI H,BUFF ; POINT TO BUFFER
SHLD RADD ; SAVE IT
RD1:
LHLD RADD ; GET BUFFER POINTER
XCHG ; INTO (D,E)
MVI C,SDMA ; CODE TO SET DMA ADDRESS
CALL BDOS ; SET DMA ADDRESS
LXI D,FCB ; POINT TO FCB
MVI C,READ ; BDOS READ CODE
CALL BDOS ; READ IN ONE SECTOR (128 BYTES)
ORA A
JNZ RD2 ; IF NON-ZERO RETURN CODE
LHLD RADD ; GET POINTER
LXI D,128
BAN D
SHLD RADD ; UPDATE IT
JMP RUI ; LOOP UNTIL DONE
RD2:
DCR A ; TEST RETURN CODE
RZ ; RETURN IF END OF FILE
LXI D,MSGE2 ; OTHERWISE GET ERROR MESSAGE
JMP EXMG ; ISSUE IT AND EXIT

; --- COMPARE MEMORY AT (H,L) TO THAT AT (D,E) FOR (C) BYTES ---

COMPARE: LDAX D ; GET BYTE
CMP M ; COMPARE
RLZ ; RETURN IF NOT EQUAL
INX H ; OTHERWISE INC. POINTERS
INX D
DCR C ; COUNT DOWN BYTES
JNZ COMPARE ; LOOP UNTIL DONE
RET

; ----- MESSAGES -----

SMGS: DB CR,LF,'--- CORVUS CONTROLLER CODE UPDATE PROGRAM ---'
DB CR,LF,( VERSION 1.2 )',CR,LF,'$

MSG1: DB CR,LF,'DO YOU WANT THE INSTRUCTIONS (Y/N) ? $

MSG2: DB CR,LF,CR,LF,'WRITE CONTROLLER CODE TO DISC (Y/N) ? $

JMSG: DB CR,LF,'IS D37 - D38 JUMPER INSTALLED (Y/N) ? $

MSG3: DB CR,LF,CR,LF
DB '---------- CONTROLLER CODE FILE HEADER MESSAGE -----------'
DB CR,LF,CR,LF,'$

MSG31: DB CR,LF,CR,LF
DB '---------------------------------------'
CRLF: DB CR,LF,'$

MSG4: DB CR,LF,CR,LF,07,'** INVALID CONTROLLER CODE FORMAT **',CR,LF,'$

MSG5: DB CR,LF,CR,LF,07,'** INVALID FILE NAME SPECIFIED **',CR,LF,CR,LF
DB 'THE PROPER CALLING SEQUENCE IS: ',CR,LF,CR,LF
DB 'A>CODE NAME.CLR',CR,LF,CR,LF
DB 'WHERE NAME.CLR IS THE FILE NAME FOR THE CONTROLLER CODE'
DB CR,LF,'$

MSG6: DB CR,LF,CR,LF,'THE CONTROLLER CODE HAS BEEN WRITTEN. NOW POWER'
DB CR,LF,'THE CORVUS DRIVE DOWN AND REMOVE THE JUMPER.',CR,LF,'$

MSG7: DB CR,LF,CR,LF,07,'** CONTROLLER CODE FILE NOT FOUND **',CR,LF,'$

MSG8: DB CR,LF,CR,LF,07,'** CONTROLLER WRITE ERROR # $

MSG9: DB 'H **',CR,LF,'$
FILE: CODE

; MSG2: DB CR,LF,CR,LF,07,’ ** DISC READ ERROR **’, CR,LF,’$’

; CHNG: DB ‘^C’, CR,LF,’$’

; MSG1: DB CR,LF,CR,LF,’ THIS PROGRAM IS USED TO UPDATE OR REPLACE’
DB CR,LF,’ CORVUS DISC CONTROLLER CODE. THIS CODE RESIDES’
DB CR,LF,’ ON PROTECTED TRACKS ON THE HARD DISC. NORMALLY’
DB CR,LF,’ THIS CODE CANNOT BE WRITTEN TO OR READ BY THE’
DB CR,LF,’ USER (EVEN ACCIDENTALLY). HOWEVER, IT CAN BE’
DB CR,LF,’ MADE ACCESSIBLE (TO WRITING) BY ADDING A JUMPER’
DB CR,LF,’ ON THE BACKPLANE PINS OF THE DRIVE (AS DESCRIBED BELOW).’
DB CR,LF,’
DB CR,LF,’ WARNING ————————————————————’, CR, F
DB CR,LF,’ DO NOT WRITE THE CODE OUT TO THE DISC WITHOUT’
DB CR,LF,’ INSTALLING THE BACKPLANE JUMPER. IF YOU DO, IT WILL’
DB CR,LF,’ BE WRITTEN OUT TO THE USER AREA OF THE DISC —’
DB CR,LF,’ OVERLAYING POSSIBLY VALUABLE USER PROGRAMS OR DATA!!’, CR,
DB CR,LF,’ TO USE THIS PROGRAM:’, CR,LF
DB CR,LF,’ 1. POWER THE CORVUS DRIVE DOWN.’
DB CR,LF,’ 2. REMOVE THE PLASTIC COVER OVER THE BACKPLANE PINS’
DB CR,LF,’ (ON THE BACK OF THE DRIVE WHERE THE POWER SUPPLY’
DB CR,LF,’ AND COMPUTER CABLES ARE ATTACHED).’
DB CR,LF,’ 3. CONNECT A JUMPER BETWEEN PINS: D37 AND D38’
DB CR,LF,’ AS ILLUSTRATED BELOW:’, CR,LF
DB CR,LF,’ HOST CONNECTOR ‘, CR,LF
DB ’+———————+———————+———————+———————+‘, CR,LF
DB ‘* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *’, CR,LF
DB ‘12 10 20 30 X40 501’, CR,LF
DB ‘* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *’, CR,LF
DB ’+———————+———————+———————+———————+’, CR,LF
DB ’ ‘, CR,LF
DB ’ ‘, CR,LF
DB ’ D37-D38 JUMPERED’, CR,LF
DB CR,LF,’ 4. POWER THE DRIVE BACK UP.’
DB CR,LF,’ 5. RUN THIS PROGRAM FROM YOUR FLOPPY CP/M’
DB CR,LF,’ WITH THE NAME OF THE CONTROLLER CODE FILE:’, CR,LF
DB CR,LF,’ A$CODE NAME.CLR’, CR,LF
DB CR,LF,’ 6. ANSWER THE PROGRAM QUESTIONS (A CONTROL-C’
DB CR,LF,’ WILL ALWAYS FORCE AN EXIT BACK TO CP/M).’
DB CR,LF,’ 7. AFTER THE CODE IS WRITTEN OUT, POWER THE DRIVE’
DB CR,LF,’ DOWN, REMOVE THE JUMPER, AND REPLACE THE COVER.’
DB CR,LF,’ NOTE: THE NEW CONTROLLER CODE WILL NOT BE’
DB CR,LF,’ ACTIVATED UNTIL THE JUMPER IS REMOVED AND THE’
DB CR,LF,’ DRIVE IS "RESET", EITHER BY THE RESET LINE OR’
DB CR,LF,’ BY A POWER DOWN/POWER UP SEQUENCE.’
DB CR,LF,’ —— IF THIS ALL GOES OK, YOU CAN NOW PROCEED TO SYSTEM’
DB CR,LF,’ RECONFIGURATION (IF NECESSARY FOR THE NEW CODE) AND OR’
DB CR,LF,’ TESTING.’, CR,LF,CR,LF,’S’

; BUFFERS AND DATA ———

; TYP: DB ‘CLR’; CP/M FILE TYPE USED FOR CONTROLLER CODE
TEST: DB CR,LF, ‘CORVUS’; EXPECTED START OF HEADER
SBUF: DS  2 ; OLD STACK POINTER
RADD: DS  2 ; RAM ADDRESS FOR DMA
DADD: DS  2 ; DISC ADDRESS
NBLKS: DS  2 ; # DISC SECTORS TO R/W
CODE: DS  2 ; BUFFER FOR SAVING POINTER
WFLG: DB  0 ; CCP OVERLAY FLAG
       DS  80 ; STACK SPACE
STACK EQU $ ;
             ORG STACK+10
BUFF EQU $ ; BUFFER FOR CONTROLLER CODE (>8K BYTES)
             END
FILE: MIRROR ASH PAGE 001

; ----- CORVUS "MIRROR" UTILITY PROGRAM ----- 
; VERSION 1.2
; BY BRK
;
; THIS PROGRAM PROVIDES THE BASIC FUNCTIONS FOR THE
; CORVUS "MIRROR" DISC BACKUP SYSTEM. IT WILL ONLY
; WORK ON SYSTEMS WITH CONTROLLER CODE VERSION > 0.
;
; ----- CP/M EQUATES ----- 
;
; BDOS EQU 05 ; BDOS ENTRY POINT
; CHIN EQU 1 ; BDOS COMMAND FOR CONSOLE INPUT
; CHOUT EQU 2 ; BDOS COMMAND FOR CONSOLE OUTPUT
; LST EQU 9 ; BDOS COMMAND FOR WRITE LIST
; RDRBUF EQU 10 ; BDOS COMMAND TO READ BUFFER
; CR EQU 0DH ; CARRIAGE RETURN
; LF EQU 0AH ; LINE FEED
;
; ----- CORVUS DISC EQUATES ----- 
;
; DATA EQU ODEM ; DATA I/O PORT
; STAT EQU DATA+1 ; STATUS I/O PORT
; DRDY EQU 1 ; MASK FOR DRIVE READY BIT
; DIFAC EQU 2 ; MASK FOR DRIVE ACTIVE BIT
; VERCOM EQU 0 ; READ VERSION # AND # OF DRIVES COMMAND
; BKUCOM EQU 8 ; MIRROR BACKUP COMMAND
; RESCOM EQU 9 ; MIRROR RESTORE COMMAND
; IDCOM EQU 10 ; MIRROR IDENT./VERIFY COMMAND
; MAXSC EQU 18936 ; MAX # OF 512 SECTORS IN DISC
; SSIZE EQU 512 ; SECTOR SIZE
;
; ORG 100H ; STANDARD CP/M TPA ORIGIN
;
; START: LXI H,0
; DAD SP ; GET STACK POINTER IN (H,L)
; SHLD SHUF ; SAVE IT
;
; -- SETUP DIRECT CONSOLE I/O JUMPS --
; LHLD 1 ; GET ADDRESS OF WARM BOOT (BIOS+3)
; LXI D,3
; DAD D ; COMPUTE ADDRESS OF CONST
; SHLD CONST+1 ; PATCH IN JUMP
; DAD D
; SHLD CONIN+1
; DAD D
; SHLD CONOUT+1
; JMP SIGNON ; SIGN ON AND START PROGRAM
;
; CONST: JMP 0 ; JUMP TO BIOS ROUTINES
; CONIN: JMP 0
; CONOUT: JMP 0
FILE: MIRROR  ASM  PAGE 002

SIGNON: LXI SP, STACK ; SETUP LOCAL STACK
LXI D, SMSG ; POINT TO MESSAGE
CALL PTMSG ; PRINT SIGN ON MESSAGE
Q1: LXI D, MSG2
MNO: CALL PTMSG ; LIST TASK MENU
MNI: LXI D, MSG3
CALL PTMSG ; ASK FOR CHOICE
MN2: CALL COMM ; GET THE TASK
MOV C, A ; MAY CONVERT ECHO TO UPPER CASE
LXI H, TSHTAB ; POINT TO TASK TABLE
MVI B, (TSHTAB - TSHTAB) / 3 ; # TASKS IN TABLE
CALL STAB ; LOOK FOR COMMAND IN TABLE
JC MN2 ; DIDN'T FIND IT, So TRY AGAIN
PUSH D ; PUT COMMAND ADDRESS ON STACK
CALL OUT ; ECHO COMMAND
LXI D, CRPLF ; CRPLF AND VECTOR TO COMMAND

; --- TASK TABLE ---

TSHTAB: EQU $ ;
DB 'L'
DW Q1
DB 'H'
DW HELP
DB 'B' ; COMMAND IDENTIFIER
DW BACKUP ; ROUTINE ADDRESS
DB 'V'
DW VERIFY
DB 'I'
DW IDENTIFY
DB 'R'
DW RESTORE
DB 'Q'
DW EXIT

TSHTBE EQU $ ; END OF TASK TABLE

; --- LIST INSTRUCTIONS COMMAND ---

HELP: LXI D, MSG1
JMP MNO

; --- BACKUP COMMAND ROUTINE ---

BACKUP: CALL INITIX ; SYNCHRONIZE AND READ VERSION $
JC MN1 ; VERSION 0, SO EXIT
MVI A, BKUROM ; GET BACKUP COMMAND
STA CMDR ; SAVE IN BUFFER
LXI H, MSG14Z
SHLD MSGPTR ; SET "CONFIDENCE MESSAGE"
CALL FILBUF ; FILL HEADER BUFFER WITH SPACES
CALL STMAX ; SET BUFFERS FOR FULL DISC SIZE
LXI D, MSG5
CALL PTMSG ; ASK IF FULL DISC
CALL YES
CNZ GTSZ ; IF NO, GET BLOCK LOCATION AND SIZE
CALL GTDRV ; GET DRIVE #
LXI H,SYSTM ; POINT TO SYSTEM TYPE
LXI D,BUF ; POINT TO BUFFER
MVI A,16 ; SIZE OF HEADER PARTS
STA PHCTR
MOV C,A ; SIZE FOR COPY
CALL COPY ; COPY TO BUFFER
XCHG
SHLD BFPTR ; SET BUFFER LOAD POINT
LXI D,MSGH
CALL PTMSG ; REQUEST HEADER DATA
LXI D,MSG9
CALL PTMSG ; ASK FOR DATE
CALL TXTIN ; GET AND SAVE IT
LXI D,MSG10
CALL PTMSG ; ASK FOR TIME
CALL TXTIN ; GET AND SAVE IT
LXI D,MSG11
CALL PTMSG ; ASK FOR NAME
CALL TXTIN
MVI A,80 ; SET NEW LINE SIZE
STA PHCTR
LXI D,MSG12
CALL PTMSG ; ASK FOR COMMENT
CALL TXTIN ; GET AND SAVE IT
LXI D,MSG13
CALL PTMSG ; ASK FOR SPEED
CALL GIPS ; GET IT
STA CKI ; SAVE IT
LXI D,MSG14
CALL PTMSG ; READY TO GO, JUST HIT CR.
BK1:
CALL CONNC ; GET CHAR.
CPI CR ; WAS IT A CR?
JNZ BK1 ; NO, SO LOOP

LXI D,MSG14Y
CALL PTMSG ; NOTIFY OF DELAY
MVI B,40 ; LONG DELAY (AT LEAST 7 SEC EVEN FOR 4MHZ 280)
BDEL:
CALL LDELAY ; WAIT FOR RECORDER TO COME UP TO SPEED
PUSH B
CALL KTST ; ISSUE MESSAGE IF KEY IS HIT
POP B
DCK B
JNZ BDEL
LXI H,MSG15 ; SET "CONFIDENCE MESSAGE"
SHLD MSGPTR

LXI H,COMD ; POINT TO START OF DATA TABLE
LXI B,SIZE+8 ; SIZE OF TABLE
CALL WFTBLK ; WRITE IT TO CONTROLLER
LXI D,MSG14X
CALL PTMSG ; "BACKUP STARTED"
CALL TURN ; WAIT UNTIL DONE
CALL WAITI ; GET ERROR TYPE
MOV C,A
CALL WAITI ; GET # OF ERRORS
MOV B,A
MOV A, C; GET TYPE BACK
ANI BOH; TEST IF SOFT
JNZ BK2; NO, SO GIVE #
MOV A, B; GET # OF ERRORS
ORA A
JNZ BK2; IF NOT ZERO
LXI D, MSG16
JMP MNO; NO ERRORS!
BK2:
LXI D, MSG17
CALL PTMSG; NOTIFY OF ERRORS
MOV A, B
CALL DECBT; GIVE HOW MANY
LXI D, MSG18; END OF MESSAGE
JMP MNO

; --- RESTORE COMMAND PROCESSOR ---
; RESTORE:
CALL INITX; SYNCHRONIZE CONTROLLER AND READ VERSION #
JC MNI; IF VERS=0
MVI A, RESCOM; GET RESTORE COMMAND
STA COMD; SET IT
LXI H, MSG42
SHLD MSGPTR; SET "CONFIDENCE MESSAGE"
CALL STMAX; SET BUFFERS TO RESTORE WHOLE DISC
LXI D, MSG40
CALL PTMSG; ASK IF WHOLE DISC
CALL YES
CNZ GTSIZ; IF NOT, GET SIZE AND LOCATION
CALL GTDRV; GET DRIVE #
LXI D, MSG41
CALL PTMSG; POSITION TAPE AND START
LXI H, COMD; POINT TO START OF BUFFER
MVI B, 7; LENGTH OF BUFFER
CALL CKSUM; CHECKSUM IT
STA CK1; SAVE CHECKSUM
LXI B, 8; LENGTH OF BUFFER TO SEND
CALL WTRBLK; SEND IT TO CONTROLLER
RST1:
CALL VLST; GET RETURN CODES AND ERRORS
JNC MNI; IF NO FATAL ERRORS
LXI H, RTRBF; POINT TO RETRY BUFFER
MVI B, 3; LENGTH OF BUFFER
CALL CKSUM; DO CHECKSUM
STA CK2; SAVE IT
CALL WTCMDS; SEND COMMANDS TO CONTROLLER
JMP RST1; DO A RETRY

; --- IDENTIFY COMMAND PROCESSOR ---
; IDENTIFY:
CALL INITX; SYNCHRONIZE CONTROLLER AND READ VERS. #
JC MNI; IF VERS=0
LXI H, MSG34
SHLD MSGPTR; SET "CONFIDENCE MESSAGE"
LXI D, MSG33
CALL PTMSG ; "POSITION TAPE ..."
LXI H, IDENT ; POINT TO COMMAND STRING
CALL WTCMD ; SEND COMMANDS TO CONTROLLER
CALL TURN ; WAIT UNTIL DONE
LXI H, BUF ; POINT TO BUFFER
SHLD BFPT ; SAVE IT
LXI B, SSIZE+4 ; SIZE OF RETURN DATA
CALL RDBLK ; READ IN DATA FROM CONTROLLER

CALL GTCHR ; GET ERROR CODE AND DISCARD
LXI D, MSG35
CALL PTMSG ; HEADER
CALL GTCHR ; GET ID #
CALL DECBT ; OUTPUT IN DECIMAL
LXI D, MSG37
CALL PTMSG
CALL GTCHR
MOV L, A ; GET LENGTH IN (H,L)
CALL GTCHR
MOV H, A
CALL DECOUT ; OUTPUT IN DECIMAL
LXI D, MSG38
CALL PTMSG ; FINISH LENGTH DESCRIPTION
MV1 A, 16
STA PRCTR ; SET STRING LENGTH
LXI D, MSG39
CALL PRTL ; LIST SYSTEM
LXI D, MSG9+2
CALL PRTL ; LIST DATE
LXI D, MSG10+2
CALL PRTL ; LIST TIME
LXI D, MSG11+2
CALL PRTL ; LIST NAME
MV1 A, 64 ; SET STRING LENGTH
STA PRCTR
LXI D, MSG12+2
CALL PRTL ; LIST COMMENT
LXI D, MSG12X+2
CALL PRTL ; REMAINDER OF COMMENT
JMP MN1

; --- VERIFY COMMAND PROCESSOR ---

VERIFY:
CALL INITX ; SYNCHRONIZE CONTROLLER AND READ VERS. #
JC MN1 ; IF VERS. #0
LXI H, MSG20
SHLD MSGPTR ; SET "CONFIDENCE MESSAGE"
LXI D, MSG19
CALL PTMSG ; "START RECORDER ...."
LXI H, VERIF ; POINT TO COMMAND STRING
CALL WTCMD ; SEND COMMANDS TO CONTROLLER
VFL:
CALL VLIST ; GET RETURN CODES AND LIST ERRORS
INC MN1 ; IF NO HARD ERRORS
LXI H, VERFL ; POINT TO RETRY-VERIFY COMMAND STRING
CALL WTCMD ; SEND COMMANDS TO CONTROLLER
JMP VFI ; LOOP TO KEEP TRYING

-------- SUBROUTINES & DATA --------

--- SEARCH TABLE FOR MATCH AND GET ASSOC. ADDRESS ---
(H,L) POINT TO TABLE TO SEARCH
(B) HAS THE # OF TABLE ELEMENTS
(C) HAS THE BYTE TO MATCH WITH
STAB: MOV A,M ; GET TABLE VALUE
       INX H ; POINT TO START OF ADDRESS
       CMP C ; IS THERE A MATCH?
       JNZ STB1 ; NO, SO CONTINUE
       MOV E,M ; GET LOWER BYTE OF ADDRESS
       INX H
       MOV D,M ; ADDRESS IN (D,E)
       RET
STB1: INX H ; SKIP OVER ADDRESS
       INX H
       DCR B ; COUNT DOWN COMMANDS
       JNZ STAB ; LOOP THRU TABLE
       STC ; NO MATCH, SO SET ERROR
       RET

--- ERROR MESSAGE LISTER FOR RESTORE AND VERIFY ---
VLST: CALL DERROR ; WAIT FOR COMMAND TO FINISH AND GET ERRORS
       MVI A,‘G’-40H ; GET A "BEL" 
       CALL PRT ; SEND TO CONSOLE
       IN STAT ; GET STATUS BYTE
       ANL DRY
       MVI A,0 ; GET O WITHOUT SETTING FLAGS
       JNZ VL1 ; IF DATA NOT READY, CONTINUE
       IN DATA ; IF AVAILABLE, GET IT
VL1: STA R2 ; SAVE IT
       MOV C,A
       LDA R1 ; GET ERROR CODE BACK
       CPI 255 ; TEST FOR MIRROR ERROR
       JNZ VL2 ; MUST BE MULTIPLE ERRORS
       LXI H,ERRTAB ; POINT TO TABLE OF ERRORS
       MVI B,(ERRTBERRTAB)/3 ; SIZE OF TABLE
       CALL STAB ; LOOK THRU TABLE
       JNC VLX ; IF MATCH, JUST LIST ERROR AND RET
       LXI D,MSG26 ; NO MATCH, SO LIST ERROR #
       CALL PRTMSG
       LDA R2
       CALL DECBT ; PRINT ERROR # IN DECIMAL
       LXI D,CRLF
VLX: JMP PRTMSG

; VL2: ANI 80H ; TEST FOR DISC ERROR
       RHZ ; IF ERROR, RETURN (NOTED ALREADY BY DERROR)
       LXI H,ERCOM ; POINT TO ERROR LIST COMMAND
       CALL WTCMD ; SEND COMMANDS TO CONTROLLER
       CALL TURN ; WAIT FOR ACCEPTANCE
LXI    D,MSG27
CALL   PTOID; PRINT ERROR TABLE HEADER
CALL   WAITI; GET # OF SOFT ERRORS IN (H,L)
MOV    L,A
CALL   WAITI
MOV    H,A
CALL   DECOUT; OUTPUT IN DECIMAL
LXI    D,MSG28
CALL   PTOID; "# OF DISC ERRORS"
CALL   WAITI; THROW THIS AWAY
CALL   DECOUT; OUTPUT IN DECIMAL
LXI    D,MSG30
CALL   PTOID; "# BLOCKS NEEDING RETRY"
CALL   WAITI
PUSH   PSM; SAVE IT ALSO
CALL   DECOUT; PRINT IT OUT
POP    PSM
ORA    A; TEST IF ZERO
JNZ    VL3; IF NOT, MUST READ MORE DATA
LXI    D,MSG31
JMP    PTOID; "ALL DATA RECEIVED"
VL3:  MOV    L,A; GET INTO (H,L)
MVI    H,0
DAD    H; DOUBLE IT (2 BYTES/BLOCK)
VL4:  CALL   WAITI; GET BYTE AND THROW AWAY
DCX    H; COUNT DOWN
MOV    A,H
ORA    L
JNZ    VL4; LOOP UNTIL DONE
LXI    D,MSG32
CALL   PTOID; "RETRY <CR>"
VL5:  CALL   CONNC
CPI    CR; IS IT A CR?
JNZ    VL5; NO, SO TRY AGAIN
LXI    D,CRLF
CALL   PTOID
STC
; NOT ERROR FOR RETRY
;

--- ERROR MESSAGE TABLE ---
;
ERRTAB:
DB     1
DW     MSG21
DB     2
DW     MSG22
DB     4
DW     MSG23
DB     7
DW     MSG24
DB     134
DW     MSG25
;
ERRTBZ: EQU 5
FILE: MIRROR  ASM  PAGE 008

; --- PRINT MESSAGE AND LIST TEXT IN BUFFER ---
;
PRTL:  PUSH   D       ; SAVE MESSAGE POINTER
    CALL PTMSG    ; PRINT MESSAGE
    LDA PRCTR     ; GET BUFFER SIZE
    MOV B,A

PTI:   CALL GTCHR  ; GET BUFFER CHARACTER
    CALL PTI      ; PRINT IT OUT
    DCR B         ; COUNT DOWN
    JNZ PTI       ; LOOP UNTIL DONE
    LXI D,CRLF
    CALL PTMSG    ; DO A CRLF
    POP D         ; GET POINTER BACK

    RET

;

GTCHR: PUSH   H
    LHLD BFPTR   ; GET BUFFER POINTER
    MOV A,H      ; GET BYTE
    INX H        ; INCREMENT POINTER
    SHLD BFPTR   ; SAVE POINTER
    POP H

    RET

;

KIST:   CALL CONST  ; TEST CONSOLE STATUS
    ORA A
    RZ
    CALL CONIN  ; OTHERWISE GET THE CHAR.
    PUSH PSW    ; SAVE CHAR.
    LHLD MSGPTR ; GET POINTER TO MESSAGE
    XCHG
    CALL PTMSG  ; PRINT IT OUT
    POP PSW     ; GET CHAR. BACK

    RET

;

--- READ IN BLOCK OF DATA FROM DISC ---
 ;
RDBLK: IN STAT  ; READ STATUS PORT
    ANI DRDY
    JNZ RDBLK
    IN DATA    ; READ BYTE FROM DISC
    MOV M,A    ; SAVE IT IN MEMORY
    INX H
    DCX B
    MOV A,B
    ORA C
    JNZ RDBLK  ; LOOP UNTIL DONE

    RET

;

--- WRITE A BLOCK OF DATA TO THE DISC ---
 ;
WTCMD: LXI B,4   ; SET SIZE FOR MIRROR COMMANDS
 ;
WTBLK: IN STAT   ; READ STATUS PORT
    ANI DRDY
    JNZ WTBLK
MOW A,M ; GET BYTE FROM MEMORY
OUT DATA ; WRITE IT TO DISC
INX H
DCE B
MOV A,B
ORA C
JNZ WTBLLK ; LOOP UNTIL DONE
RET

; DERROR: CALL TURN ; TURN AROUND BUS
DERR1: CALL WAIT1 ; WAIT FOR ERROR BYTE
MOV B,A ; SAVE BYTE
STA R1 ; SAVE IN BUFFER ALSO
CPI 255 ; TEST FOR MIRROR ERROR
RZ ; RETURN FOR LATER LISTING
ANI 80H ; LOOK FOR FATAL ERRORS
RZ ; OK, SO RETURN
PUSH B ; SAVE ERROR
LXI D,MSGE ; ERROR, SO ISSUE MESSAGE
CALL PTMSG
POP PSW ; GET ERROR BYTE BACK IN ACC
CALL HEXOT ; OUTPUT IN HEX
LXI D,MSGE1
CALL PTMSG
RET

; TURN: CALL KST ; TEST FOR KEY DOWN
IN STAT
ANI DIFAC OR DRDY ; TEST IF INACTIVE AND READY
JNZ TURN
HVI B,15 ; GOOD AT 4MHZ ALSO
CALL DELAY
IN STAT
ANI DIFAC OR DRDY ; TEST IF INACTIVE AND READY
JNZ TURN
RET

; DELAY: DCR B
JNZ DELAY
RET

; ---- LONG DELAY ROUTINE ----
LDELAY: PUSH B
LXI B,41665 ; SET FOR 0.5 SEC (2 MHZ 8080A)
LDELI: DCE B
MOV A,B
ORA C
JNZ LDELI ; LOOP UNTIL DONE
POP B
RET

WAIT1: IN STAT ; READ STATUS PORT
ANI DRDY ; LOOK AT READY LINE
JNZ WAIT1 ; LOOP UNTIL READY
IN DATA ; READ BYTE FROM DISC
RET

; WAITO: PUSH PSW ; SAVE COMMAND
IN STAT ; READ STATUS PORT
ANI DRDY ; LOOK AT READY LINE
JNZ WAITO+1 ; LOOP UNTIL READY
POP PSW
OUT DATA ; WRITE BYTE TO DISC
RET

; --- INITIALIZE CONTROLLER ---

INIT: MVI A,OFFH ; GET AN INVALID COMMAND
OUT DATA ; SEND IT TO CONTROLLER
MVI B,150 ; SET FOR LONG DELAY
CALL DELAY
IN STAT
ANI DIFAC ; LOOK AT DRIVE ACTIVE BIT
JNZ INIT ; LOOP UNTIL NOT ACTIVE
CALL WAITI ; GET ERROR CODE
CPI 8FH ; CHECK RETURN CODE
JNZ INIT ; IF NOT RIGHT, TRY AGAIN

; TEST CONTROLLER CODE VERSION

MVI A,VERCOM ; GET COMMAND TO READ VERSION # AND # OF DRIVES
CALL WAITO ; SEND IT
CALL TURN ; WAIT FOR ACCEPTANCE
CALL WAITI ; GET ANSWER
ANI 0FH ; MASK OUT # OF DRIVES
RET

INITX: CALL INIT ; INITIALIZE AND TEST VERS. #
RNZ ; RETURN IF #>0
LKX D,MSG4
CALL PTMSG ; ISSUE ERROR MESSAGE
STC ; SET ERROR CONDITION
RET

; --- MESSAGE PRINT ROUTINE---

PTMSG: MVI C,LST ; CP/H WRITE LIST COMMAND
CALL BDOS ; EXECUTE BDOS COMMAND
ORA A ; INSURE CARRY IS CLEARED
RET

; --- OUTPUT BYTE IN ACC IN HEX ---

HEXOUT: PUSH PSW ; SAVE BYTE
RRC
RRC
RRC
RRC
CALL HEXB ; OUTPUT UPPER NIBBLE IN HEX
FILE: MIRROR ASH PAGE 011

HEX8:
PUSH PSW
POP PSW
; GET BYTE BACK
ANI OFH
; MASK OFF UPPER NIBBLE
ADI '0'
; ADD ASCII BIAS
GPI '9' + 1
; TEST IF NUMERIC
JC PRT
; YES, SO DO IT
ADI 7
; NO, SO ADD BIAS FOR A-F
PRT: MOV C, A
; SETUP FOR OUTPUT
COUT: PUSH PSW
; BUFFERED CONSOLE OUTPUT
PUSH H
PUSH D
PUSH B
MOV E, C
MVI C, CHOUT
; BDOS CHAR. OUTPUT COMMAND
CALL BDOS
POP H
POP D
POP B
POP PSW
RET

; --- OUTPUT (H, L) IN DECIMAL ---

DECOUT: LXI D, -10000
; SET TO SUBTRACT 10000
MVI B, '0'
; SET TO SUPPRESS LEADING ZEROS
CALL DEC2
; OUTPUT FIRST CHAR.
LXI D, -1000
; SET TO SUBTRACT 1000
CALL DEC2
; OUTPUT SECOND CHAR.
DEC4: LXI D, -100
; SET TO SUBTRACT 100
CALL DEC2
; OUTPUT THIRD CHAR.
LXI D, -10
; SET TO SUBTRACT 10
CALL DEC2
; OUTPUT FOURTH CHAR.
MVI B, 0
; ALLOW LEADING ZERO
LXI D, -1
; SET TO SUBTRACT 1
DEC2: MVI C, '0' + 1
; SET CHAR. COUNT
DEC3: SHLD DECBUF
; SAVE REMAINDER
INR C
; INC. ASCII CHAR. COUNTER
DAD D
; DO SUBTRACTION
JC DEC3
; LOOP UNTIL UNDERFLOW
LHLD DECBUF
; GET LAST REMAINDER
MOV A, C
; GET CHAR. COUNTER
CMP B
; TEST FOR ZERO SUPPRESS
JZ SPACE
; ISSUE SPACE IF ZERO SUPPRESS IS ON
MVI H, 0
; CLEAR ZERO SUPPRESS FLAG
JMP COUT
; OUTPUT CHAR.
SPACE: MVI C, '
; SEND ASCII SPACE TO CONSOLE
JMP COUT

; -- OUTPUT BYTE IN DECIMAL --

DECBT: PUSH H
PUSH D
PUSH B
MOV L, A
; SAVE BYTE IN (H, L)
MVI H, 0
MVI B, '0'
; SET TO SUPPRESS LEADING ZEROS
CALL DEC4
POP B
; -- TWO BYTE DECIMAL INPUT ROUTINE --

INDEC: LXI H,0 ; CLEAR CONVERSION REGISTER
IN1: PUSH H
CALL CONNC ; GET CHARACTER
POP H
CPI \ ; IS IT A SPACE?
JZ IN1 ; IGNORE IT
CPI CR ; IS IT A CR?
RZ ; YES, SO RETURN
CALL COUT ; ECHO CHAR.
SUI '0' ; REMOVE ASCII BIAS
RC ; RETURN IF ERROR
CPI 10 ; TEST IF TOO BIG
CMC
RC ; RETURN IF ERROR
MOV E,L ; GET COPY OF (H,L) IN (D,E)
MOV D,H
DAD H ; MULTIPLY BY 5
DAD H
DAD D
DAD H ; NOW 10 X STARTING VALUE
MOV E,A
MVI D,0
DAD D ; ADD IN NEW UNITS DIGIT
PUSH H ; SAVE IT
LXI D,MAXLOC ; GET MAX. DISC ADDRESS
DAD D
POP H
JNC IN1 ; IF OK, GET MORE DIGITS
RET ; RETURN IF ERROR

; -- YES FUNCTION --

YES: CALL CONNC ; GET CONSOLE CHAR.
CPI 'Y' ; IS IT A Y?
JZ YES1 ; IS IT A N?
CPI 'N' ; IS IT A M?
JNZ YES ; IF NEITHER, KEEP TRYING
INR A ; SET N STATUS
YES1: PUSH PSW ; SAVE FLAGS
CALL CONOUT ; OUTPUT TO CONSOLE
POP PSW ; RESTORE FLAGS
RET

CONNC: CALL CONIN ; GET CHAR. FROM CONSOLE
MOV C,A ; SAVE FOR ECHO
CPI 60H ; IS IT LOWER CASE?
JC CON1 ; NO, SO CONTINUE
ANI 5FH ; YES, SO MASK TO UPPER CASE
CON1: CPI 'C'-'40H ; IS IT A CONTROL-C?
RNZ ; NO, SO RETURN

CTC: LXI D,CMMSG ; POINT TO CONTROL-C MESSAGE
FILE: MIRROR ASH PAGE 013

EXMG:
CALL PTMSG ; ISSUE MESSAGE
EXIT:
LXI D,CRLF
CALL PTMSG ; ISSUE A CRLF
LHLD SBUF ; GET OLD STACK POINTER
SPHL ; SET STACK
RET ; BACK TO CP/M

--- ASK FOR AND GET DRIVE # ---

GTDRV:
LXI D,DMSG
CALL PTMSG ; ASK FOR DRIVE #
GTDRV1:
CALL CONNC ; GET CONSOLE CHAR.
CPI '1'
JC GTDRV1 ; IF INVALID, TRY AGAIN
CPI '4' +1
JNC GTDRV1
SUI '0' ; REMOVE ASCII BIAS
STA DRIVE ; SAVE IT
CALL COUT ; ECHO IT
LXI D,CRLF
JMP PTMSG

--- ASK FOR AND GET DISC BLOCK LOCATION AND SIZE ---

GTSIZ:
LXI D,MSG6
CALL PTMSG ; ASK FOR STARTING BLOCK #
CALL INDEC ; GET IT
JC GTSIZ ; IF ERROR, TRY AGAIN
SHLD BKSTR ; SAVE IT IN BUFFER
GTSZI:
LXI D,MSG7
CALL PTMSG ; ASK FOR LENGTH
CALL INDEC ; GET IT
JC GTSZI ; IF ERROR, TRY AGAIN
MOV A,H
ORA L ; IS IT ZERO?
JZ GTSZI ; YES, SO TRY AGAIN
SHLD BLEN ; SAVE IT
XCHG
LHLD BKSTR ; GET STARTING LOC. BACK
DAD D ; FIND ENDING LOC.
LXI D,-MAXSC
DAD D ; CHECK IF TOO BIG
MNC
LXI D,BMSG
CALL PTMSG ; PRINT ERROR MESSAGE
JMP GTSIZ ; TRY AGAIN

--- GET SPEED OF BACKUP (FAST OR NORMAL) ---

GTSPD:
CALL CONNC ; GET CHAR.
CPI 'F' ; WAS IF FAST?
MVI B,0 ; # FOR FAST
JZ GTSI ; YES
CPI 'N' ; WAS IT NORMAL?
MVI B,1 ; # FOR NORMAL
FILE: MIRROR ASM PAGE 014

JNZ CTSPD ; IF NO MATCH, TRY AGAIN
GTS1: CALL COUT ; ECHO KEY HIT
       MOV A,B ; GET #
       RET

; --- SET BUFFERS FOR WHOLE DISC SAVE/RESTORE ---

STMAX: LXI H,MAXSC ; GET # 512 BLOCKS ON DISC
       SHLD BLEN
       LXI H,O ; GET STARTING DISC ADDRESS
       SHLD BKSTR
       RET

; --- BLOCK CHECKSUM ROUTINE ---

CKSUM: PUSH H ; SAVE POINTER
       XRA A ; INITIALIZE
       MOV C,M ; GET BYTE
       ADD C ; DO CHECKSUM
       INX H ; POINT TO NEXT LOC.
       DCR B ; COUNT DOWN
       JNZ CKSUM+2 ; LOOP UNTIL DONE
       CHA ; FIND NEGATIVE AND RETURN
       INR A
       POP H
       RET

; --- INPUT TEXT LINE AND SAVE IN DISC BUFFER ---

TXTIN: LDA PRCTR ; GET BUFFER SIZE
       LXI D,ISBN ; POINT TO TEXT BUFFER
       STAX D ; SAVE MAX SIZE (FOR CP/M FUNCTION)
       MVI C,RDBUF ; GET CP/M BUFFER READ COMMAND
       CALL BDOS ; INPUT TEXT STREAM
       LHLBD BFPTER ; GET BUFFER POINTER
       PUSH H ; SAVE IT
       LDA PRCTR ; GET MAX TEXT BLOCK SIZE
       MOV E,A ; GET INTO (D,E)
       MVI D,0
       DAD D ; COMPUTE NEW POINTER
       SHLD BFPTER ; SAVE IT
       POP D ; GET BACK DESTINATION ADDRESS
       LXI H,ISBN+1 ; POINT TO BUFFER COUNTER
       MOV A,M ; GET IT
       ORA A ; IS IT ZERO?
       RZ ; YES, SO FINISH
       MOV C,M ; NO, SO GET AS COUNTER
       INX H ; POINT TO START OF TEXT
       COPY: MOV A,M ; GET SOURCE BYTE
       STAX D ; SAVE COPY AT DESTINATION
       INX H
       INX D
       DCR C ; COUNT DOWN # TO COPY
       JNZ COPY ; LOOP UNTIL DONE
       RET
FILE: MIRROR ASH PAGE 015

; --- BUFFER FILL ROUTINE ---
; FILBUF: LXI B, SSIZE+4 ; SET SIZE
   LXI H, BUF ; LOCATION OF BUFFER
FILL:  MVI M, ' ' ; FILL WITH SPACES
   INX H
   DCX B
   MOV A, B
   ORA C
   JNZ FILL ; LOOP UNTIL DONE
   RET

; ;
; ---- MESSAGES ----
; SMSG: DB CR,LF,' --- CORVUS MIRROR UTILITY ---'
   DB CR,LF,' ( VERSION 1.2 ) ','CR,LF,'$'
; MSG2: DB CR,LF,' --- MIRROR MENU --- ','CR,LF
   DB CR,LF,'L: LIST THIS MENU'
   DB CR,LF,'H: LIST HELP DATA'
   DB CR,LF,'B: BACKUP'
   DB CR,LF,'V: VERIFY'
   DB CR,LF,'I: IDENTIFY'
   DB CR,LF,'R: RESTORE'
   DB CR,LF,'Q: QUIT'
   DB CR,LF,'$'
; MSG3: DB CR,LF,' TASK (L TO LIST) : $'
; MSG4: DB CR,LF,07
   DB ' --> THIS FEATURE IS NOT AVAILABLE UNDER VERS. 0 CONTROLLER CODE'
   DB CR,LF,'$'
; MSG5: DB CR,LF,CR,LF,07,' ** DISC R/W ERROR # $'
; MSGE1: DB 'H **',CR,LF,'$
; DMSG: DB CR,LF,' CORVUS DRIVE # (1-4) ? $
; CHSG: DB 'C',CR,LF,'$
; RMSG: DB CR,LF,07,' -- THIS WOULD EXCEED DISC SIZE -- ',CR,LF,'$
; MSG5: DB CR,LF,' BACKUP ENTIRE CORVUS DISC (Y/N) ? $
; MSG6: DB CR,LF,' STARTING DISC BLOCK # ? $
; MSG7: DB CR,LF,' NUMBER OF BLOCKS ? $
; MSG8: DB CR,LF,07,' ** THIS WOULD EXCEED DISC SIZE ** ',CR,LF,'$
; MSGH: DB CR,LF,' --- ENTER TAPE FILE HEADER INFORMATION --- ',CR,LF
   DB '$'
; MSG9: DB CR,LF,' DATE : $

FILE: MIRROR ASH PAGE 016

MSG10: DB CR,LF, ' TIME : $'
MSG11: DB CR,LF, ' NAME : $'

; MSG12: DB CR,LF, ' COMMENT : $'
; MSG12X: DB CR,LF, '$'
; MSG13: DB CR,LF, ' NORMAL OR FAST FORMAT (N/F) ? $'
; MSG14: DB CR,LF, ' STARTUP RECORDER AND PRESS RETURN $'
; MSG14X: DB CR,LF, ' >> BACKUP HAS STARTED <<', CR,LF, '$'
; MSG14Y: DB CR,LF
; MSG14Z: DB CR,LF, ' WAITING FOR RECORDER TO SPEED UP ...', CR,LF, '$'
; MSG15: DB CR,LF, ' BACKUP IN PROGRESS ...', CR,LF, '$'
; MSG16: DB CR,LF, ' BACKUP DONE -- NO ERRORS', CR,LF, '$'
; MSG17: DB CR,LF, ' THERE WERE $'
; MSG18: DB ', DISC READ ERRORS DURING BACKUP $', CR,LF
; MSG19: DB CR,LF, ' START RECORDER AT BEGINNING OF IMAGE', CR,LF
; MSG20: DB CR,LF, ' VERIFY IN PROGRESS ...', CR,LF, '$'
; MSG21: DB CR,LF, ' IMAGE ID NOT EQUAL TO $', CR,LF, '$'
; MSG22: DB CR,LF, ' MIRROR ERROR 2', CR,LF, '$'
; MSG23: DB CR,LF, ' IMAGE SIZE IS WRONG FOR THIS RESTORE', CR,LF, '$'
; MSG24: DB CR,LF, ' TIMEOUT - VIDEO NOT RECEIVED', CR,LF, '$'
; MSG25: DB CR,LF, ' VIDEO INTERRUPTED IN MIDDLE OF IMAGE', CR,LF, '$'
; MSG26: DB CR,LF, ' MIRROR ERROR $'
; MSG27: DB CR,LF, ' --- ERROR STATISTICS ---', CR,LF, CR,LF
   DB ' # SOFT ERRORS :$'
; MSG28: DB CR,LF, ' # DISC ERRORS : $'
; MSG30: DB CR,LF, ' # OF BLOCKS NEEDING RETRYS : $'
; MSG31: DB CR,LF, CR,LF, ' ALL DATA RECEIVED', CR,LF, '$'
; MSG32: DB CR,LF, 07, ' -- RETRY NEEDED --
   DB CR,LF, ' START RECORDER AT BEGINNING OF IMAGE -- PRESS RETURN $'
; MSG33: DB CR,LF, ' POSITION TAPE AND START PLAYBACK', CR,LF
; MSG34: DB CR,LF, ' SEARCHING FOR IMAGE HEADER ...', CR,LF, '$'
MSG35: DB CR,LF,' --- IMAGE RECORDED FROM CORVUS DRIVE ---',CR,LF
   DB CR,LF,' IMAGE ID :$',CR,LF
MSG37: DB CR,LF,' IMAGE LENGTH :$',CR,LF
MSG38: DB ' BLOCKS ',CR,LF,'$
MSG39: DB CR,LF,' SYSTEM :$
MSG40: DB CR,LF,' RESTORE ENTIRE DISC (Y/N) ?$
MSG41: DB CR,LF,' POSITION TAPE AND START PLAYBACK ',CR,LF
MSG42: DB CR,LF,' RESTORE IN PROGRESS ...',CR,LF,'$

SYSTM: DB '1IS

MSG1: DB CR,LF,' THIS PROGRAM PROVIDES THE BASIC CONTROL FUNCTIONS'
DB CR,LF,' FOR THE CORVUS "MIRROR" DISC BACKUP SYSTEM. IT WILL'
DB CR,LF,' ONLY WORK ON SYSTEMS WITH CONTROLLER CODE VERSION > 0.'
DB CR,LF,' FUNCTIONS PROVIDED ARE: ',CR,LF
DB CR,LF,' B: BACKUP'
DB CR,LF,' COPY A CONTIGUOUS SECTION OF INFORMATION ON THE'
DB CR,LF,' CORVUS DRIVE ONTO A VIDEO TAPE FILE.'
DB CR,LF,' V: VERIFY'
DB CR,LF,' RE-READ A VIDEO TAPE FILE AND VERIFY THAT IT HAS'
DB CR,LF,' BEEN RECORDED CORRECTLY. THIS IS DONE BY TESTING'
DB CR,LF,' THE CRC (A FORM OF CHECKSUM) OF EACH RECORD.'
DB CR,LF,' I: IDENTIFY'
DB CR,LF,' READ THE HEADER OF A VIDEO TAPE FILE AND LIST IT'
DB CR,LF,' ON THE CONSOLE.'
DB CR,LF,' R: RESTORE'
DB CR,LF,' COPY A VIDEO TAPE FILE BACK TO THE CORVUS DRIVE.'
DB CR,LF,' IT NEED NOT BE RESTORED TO THE SAME PLACE IT WAS'
DB CR,LF,' COPIED FROM.',CR,LF
DB CR,LF,' H: RETRY'
DB CR,LF,' THIS FUNCTION IS BUILT IN TO THE VERIFY AND RESTORE'
DB CR,LF,' FUNCTIONS. A RETRY WILL BE REQUESTED IF THE REDUNDANCY'
DB CR,LF,' BUILT INTO THE "MIRROR" RECORDING FORMAT WAS NOT'
DB CR,LF,' SUFFICIENT TO RECOVER FROM AN ERROR DETECTED IN ONE OR'
DB CR,LF,' MORE TAPE RECORDS. IN THIS CASE, THE ERROR STATISTICS'
DB CR,LF,' WILL SHOW HOW MANY BLOCKS NEED RETRYS (NOTE: IF THIS'
DB CR,LF,' NUMBER IS ZERO THEN ALL OF THE DATA WAS RECOVERED).'
DB CR,LF,' A CONTROL - C ISSUED IN RESPONSE TO A PROMPT WILL CAUSE'
DB CR,LF,' AN EXIT BACK TO CP/M. A NON DECMIAL INPUT, IN RESPONSE'
DB CR,LF,' TO A Prompt REQUESTING A NUMBER, WILL CAUSE A REPEAT OF'
DB CR,LF,' THE QUESTION ( CONTROL - C WILL ALWAYS CAUSE AN EXIT).'
DB CR,LF,' THE ONLY NUMERICAL INPUTS REQUIRED ARE ALL IN DECIMAL.'
DB CR,LF,' THE BACKUP AND RESTORE COMMANDS MAY ASK FOR THE'
DB CR,LF,' " STARTING DISC BLOCK # " AND THE " # OF BLOCKS "
DB CR,LF,' (IF YOU ARE NOT SAVING OR RESTORING AN ENTIRE DISC).
DB CR,LF,' THIS REFERS TO THE ACTUAL INTERNAL ORGANIZATION OF
DB CR,LF,' THE DRIVE - WHICH USES 512 BYTE SECTORS (BLOCKS).
DB CR,LF,' THE RELATION BETWEEN THE BLOCK ADDRESS (0 - 18935)
DB CR,LF,' AND THE USUAL 128 BYTE DISC ADDRESS (0 - 75743).
DB CR,LF,' IS SIMPLEx:CR,LF
DB CR,LF,' DISC ADDRESS (128 BYTE) = 4 X BLOCK ADDRESS,CR,LF
DB CR,LF,' THIS MAY CAUSE A SLIGHT PROBLEM IF YOU WANT TO SAVE
DB CR,LF,' OR RESTORE DISC DATA AT DISC ADDRESSES (128 BYTE).
DB CR,LF,' THAT ARE NOT DIVISIBLE BY 4.
DB CR,LF,'$'

; ----- BUFFER AND DATA -----

SBUF: DS 2         ; OLD STACK POINTER
DADD: DS 2         ; DISC ADDRESS
BFPTR: DS 2        ; BUFFER POINTER
PRCTR: DS 1        ; COUNTER FOR BUFFER ROUTINES
DECBUF: DS 2      ; BUFFER FOR DECOUNT ROUTINE
R1:   DS 1        ; BUFFER FOR ERROR CODE

MSGPTR: DS 2      ; POINTER TO MESSAGE FOR KTST ROUTINE

IDENT: DB 10,0,1,0 ; COMMAND SEQ. FOR IDENTIFY COMMAND

VERIF: DB 10,1,1,0 ; COMMAND SEQ. FOR VERIFY COMMAND

VERFI: DB 10,6,1,0 ; COMMAND FOR RETRY, VERIFY

ERCOM: DB 10,2,0,0 ; LIST ERRORS COMMAND

RTRBF: DB 10      ; RETRY COMMAND
DB 3
R2:  DS 1         ; # OF ERRORS (BUFFER)
CK2: DS 1         ; BUFFER FOR CHECKSUM
DS 5              ; EXTRA SPACE

TXBUFF: DS 200H    ; TEXT BUFFER AREA

COMD: DB 9        ; BACKUP COMMAND

DRIVE: DS 1       ; BUFFER FOR DRIVE #
ID:  DB 1         ; STANDARD ID #
BLEN: DS 2        ; BUFFER FOR LENGTH (IN 512 BYTE BLOCKS)
BKSTRT: DS 2      ; BUFFER FOR STARTING BLOCK #
CK1:  DS 1        ; BUFFER FOR CHECKSUM
BUF:  DS 520      ; HEADER BUFFER

DS 80            ; STACK SPACE

STACK EQU $       ; END
**PSEUDO DRIVE SIZES AND LOCATIONS**

FOR USE WITH

THE MIRROR

THE CP/M MIRROR UTILITY PROVIDED BY CORVUS ALLOWS THE USUAL MIRROR FUNCTIONS (BACKUP, IDENTIFY, VERIFY, AND RESTORE) TO BE APPLIED TO ANY CONTIGUOUS SECTION OF THE CORVUS DRIVE (IN TERMS OF 512 BYTE BLOCK ADDRESS AND LENGTH). BECAUSE CP/M ALLOWS SUCH TOTAL FLEXIBILITY FOR THE CHOICE OF DISC LAYOUT, WE CHOSE TO ALLOW DIRECT SPECIFICATION OF THE BLOCK ADDRESS AND LENGTH (RATHER THAN ASSUME A SPECIFIC LAYOUT OF THE PSEUDO DRIVES). HOWEVER, TO BE USEFUL WITH THE MIRROR, ONE SHOULD DESIGN THE ARRANGEMENT OF THE PSEUDO DRIVES SO THAT EACH DIRECTORY STARTS ON A 512 BYTE BLOCK BOUNDARY. ALSO, IT IS USEFUL TO HAVE AT LEAST TWO PSEUDO DRIVES OF EQUAL SIZE FOR EACH SIZE CHOSEN. THIS ALLOWS FOR COPYING AND RESTORING OF FILES BETWEEN BACKUP TAPE FILES AND ONE OF THE OTHER PSEUDO DRIVES OF EQUAL SIZE. FOR REFERENCE, WE HAVE LISTED THE BLOCK ADDRESS AND LENGTHS FOR THE CP/M 1.4 AND 2.0 INTERFACE ROUTINES DISTRIBUTED WITH THIS RELEASE OF OUR SOFTWARE.

<table>
<thead>
<tr>
<th>SYSTEM</th>
<th>DRIVE</th>
<th>STARTING BLOCK NUMBER</th>
<th>BLOCK LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP/M 2.0</td>
<td>A</td>
<td>18</td>
<td>9440</td>
</tr>
<tr>
<td></td>
<td>B</td>
<td>9486</td>
<td>9440</td>
</tr>
<tr>
<td></td>
<td>C</td>
<td>18</td>
<td>9440</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>9486</td>
<td>9440</td>
</tr>
<tr>
<td>CP/M 1.4</td>
<td>C</td>
<td>0</td>
<td>1024</td>
</tr>
<tr>
<td></td>
<td>D</td>
<td>1024</td>
<td>1024</td>
</tr>
<tr>
<td></td>
<td>E</td>
<td>2048</td>
<td>1024</td>
</tr>
<tr>
<td></td>
<td>F</td>
<td>3072</td>
<td>1024</td>
</tr>
<tr>
<td></td>
<td>G</td>
<td>4096</td>
<td>1024</td>
</tr>
<tr>
<td></td>
<td>H</td>
<td>5120</td>
<td>1024</td>
</tr>
<tr>
<td></td>
<td>I</td>
<td>6144</td>
<td>1024</td>
</tr>
<tr>
<td></td>
<td>J</td>
<td>7168</td>
<td>1024</td>
</tr>
<tr>
<td></td>
<td>K</td>
<td>8192</td>
<td>1024</td>
</tr>
<tr>
<td></td>
<td>L</td>
<td>9216</td>
<td>1024</td>
</tr>
<tr>
<td></td>
<td>M</td>
<td>10240</td>
<td>1024</td>
</tr>
<tr>
<td></td>
<td>N</td>
<td>11264</td>
<td>7648</td>
</tr>
</tbody>
</table>