creagx operating system

PROJECT WORK

Done by

KIRON .K. ABRAHAM
S. SATHISH KUMAR
D. SENTHIL KUMAR
H. THIRUKKUMARAN
K. VINOD KUMAR

P. 660

Under the guidance of

Mrs. R. KALAISELVI B.E.

P. 660

submitted in partial fulfillment of the requirements for the degree of

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE

Of the Bharathiar University, Coimbatore

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

COIMBATORE 641 006

MARCH-2002

<u>CERTIFICATE</u>

Department of Computer Science and Engineering
KUMARAGURU COLLEGE OF TECHNOLOGY
COIMBATORE 641 006

This is to certify that the Project Report entitled

creagx operating system

is the bonafide record of work done by

KIRON .K. ABRAHAM	**(9827K0184)**
S. SATHISH KUMAR	**(9827K0212)**
D.SENTHIL KUMAR	**(9827K0215)**
H. THIRUKKUMARAN	**(9827K0221)**
K. VINOD KUMAR	**(9827K0224)**

In the partial fulfillment of the requirements for the degree of
BACHELOR OF ENGINEERING
In
COMPUTER SCIENCE
Of the Bharathiar University, Coimbatore

[illegible] 12/3/2002
Mrs. R. Kalaiselvi
Project Guide

[illegible]
Dr. S. Thangasamy
Head of the Department

Submitted for the Viva Voce held on

14th March, 2002

[illegible]
Internal Examiner

External Examiner

making the dusk a dawn...

creagx operating system

A dedication...

to our beloved parents, brothers and sisters
for the unconditional love and affection

to Kumaraguru College of Technology
for opening the doors of technical
education

to all our honourable teachers
for the knowledge acquired

to our respected guide
for her intuition, advice and inspiration

to all our friends and loved ones
for their unfailing support and care.

❖ ACKNOWLEDGEMENT

ACKNOWLEDGEMENT

We hereby express our sincere gratitude to our beloved principal. **Dr. K. K Padmanabhan** for providing all the facilities needed for the completion of this project.

We are extremely grateful to our Head of the Department, **Dr. S. Thangasamy Ph.D.** for being the corner stone of our project. His valuable advices and supervision only brought this project into existence.

We extend our heartfelt thanks to our Assistant Professor, **Mrs. S. Devaki M. S.**, for providing us the complete utilization of the computer science laboratory whenever needed. Also, she helped us with some valuable suggestions, which actually made our project worth its salt.

Our project Guide, **Mrs. R. Kalaiselvi**, needs to be expressed a huge lump of thanks for the strain she had to take in helping us. She was always beside us providing quick personal and bookish references during the developing stages of this project. Without her strong support and cooperation, the project would have remained a dream.

We extend a very special gratitude to **Mr. Shoban Jeyaraj** of Think Business Networks (TBN), Coimbatore for providing us the required utilities and tools for framing the project in a very standard way. Even in his tight schedule, he was kind enough to allocate a few hours for discussions regarding our project.

In addition, we are grateful to all the teaching as well as non-teaching staffs of the Department of Computer Science and Engineering who have helped us in one context or other.

Finally, we quote a note of thanks to all our friends for their eternal inspiration and encouragement.

SYNOPSIS

SYNOPSIS

'creagx' operating system blooms in the world of many familiar operating systems. But, most of them either deliberately or accidentally missed some interesting features. This operating system is a joint venture to highlight these limitations. 'creagx' operating system is developed right from the scratch and hence it does not require any software substratum. The main feature of this operating system is the utilization of Processor Serial Number (PSN), which is an enhanced architecture for the Pentium III processor. 'creagx' is a 16bit operating system with open source code. In effect, our operating system concentrates on

- Ending software piracy to some extent
- Designing a processor specific operating system.
- A unique FAT relocation.

The File Allocation Table entry is stored in inner to out fashion on the disk where as the file storage is performed from the outer to the inner track. 'creagx', although Pentium III specific, works on all the X86 processors. Slight enhancements are also performed in the booting process. Device drivers for most of the prevailing devices are implemented. A complete set of command is provided in this operating system. Also, utilities are provided to enhance the operating system standards. A well-formulated help is also available.

The 'creagx' operating system is completely done using Net Wide Assembler (NASM), which is provided by Mr. Rob Anderton. The complete source code is kept open welcoming any further enhancements. For enhancing reliability, each phase of the project is verified using sophisticated tools.

INDEX

INDEX

INTRODUCTION

1. INTRODUCTION

Operating Systems, as H.M Dietel states, are primarily resource managers. It forms a bridge between the user and the hardware. Today's world has a wide exhibition of operating systems; each one having a fair collection of features. However, all of them appear to share some limitations. In the efforts to point out these drawbacks, bloomed 'creagx' operating system.

1.1 EXISTING OPERATING SYSTEMS

Existing operating systems all concentrate on the variety of applications. The existing software scenario demands a lot of authentication factors to be implemented. The e-commerce and e-business are right at the throne of the current market.

The well prevailing operating systems provide no software authentication facilities or security features. The applications through internet depend on the IP address to some extent. But, each time you are logging in, the IP address will be changed. Hence, the reliability of the authentication feature using IP address is worth questioning.

It will be extremely useful if you can utilize the hardware feature of a system for implementing the security concepts. Indeed, there is an architecture in Pentium III which remained in dusk for the developers. This feature is called the Processor Serial Number. Since no operating system utilized it, the Intel Corporation neglected this architecture in their advanced processor, the Pentium IV.

1.2. 'creagx' AND ITS ADVANTAGES

'creagx' provides an access to the Processor Serial Number, which can be utilized to end software piracy. Also small tokens of improvements are implemented in each phase of development. The key features of 'creagx' are explained below.

The PSN Utility:

'creagx' is designed specifically for Pentium III processor, but it will work on all processors belonging to the x86 family. Processor Serial Number is made accessible in this operating system, thereby opening the possibility of a hardware oriented security feature. We have used the Processor Serial Number (PSN), which is a specific feature of Pentium III processor as a utility accounting for its processor specific structure.

Unique Booting:

Any operating system on booting, checks the word aa55h at the location 07DFEh. This 16-bit word is the boot signature. The signature is used by BIOS to ensure that the sector contains a valuable boot record. All the operating systems do not mask this signature and as a result on continuous improper shut down, the signature gets corrupted and the operating system asks the user regarding the location of its primary file.

'creagx' operating system masks the signature bit thereby avoiding the corruption of the boot signature. Hence, on any number of improper shutdowns, the booting proceeds normally.

Unique File Structure:

'creagx' senses the need for altering the ordinary file storage system. In order to avoid the relocation of the set of files on each FAT entry, the FAT is organized to exist from the inner track of the disk to the outer. The storage of files is performed in the opposite direction. If the difference between the FAT and the stored files is reduced to a single track, the user is warned regarding low disk space and for safety measures, the file is discarded.

Adding to these features, small tokens of innovations are implemented in areas like display of prompt, file creation and commands. The open source code is another distinctive feature of 'creagx' operating system.

❖ SYSTEM REQUIREMENTS

2. SYSTEM REQUIREMENTS

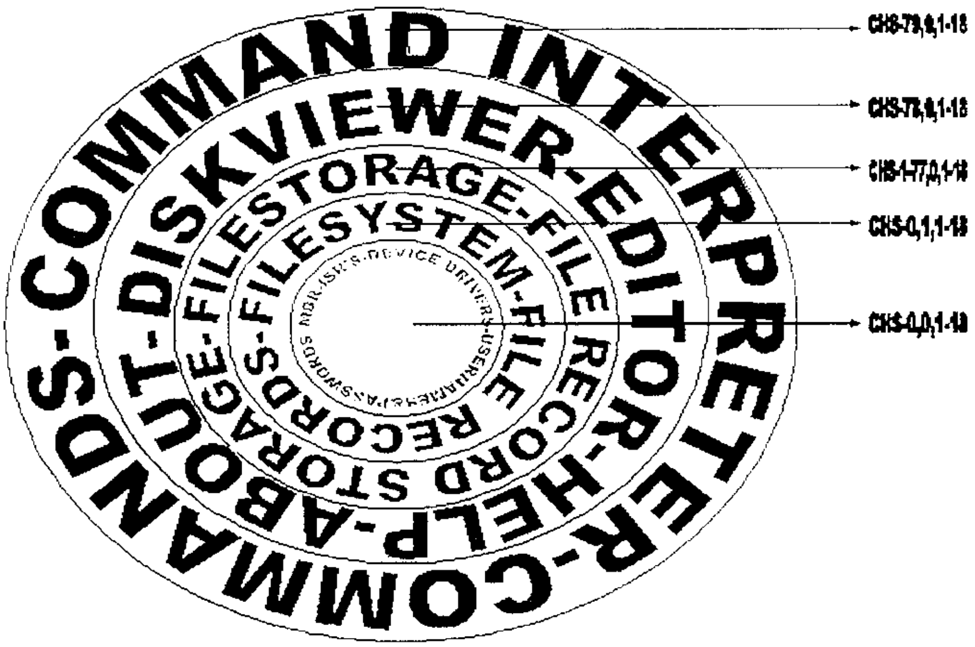
'creagx' operating system has been designed from the scratch itself. Hence, it does not need any substratum for development. This operating system is developed in Net Wide Assembler (NASM). The assembler is composed by Mr. Rob Anderton. The basic assembly language programming knowledge is the key resource required to work in NASM. It resembles the TURBO C assembler in many respects thereby making the working easy. In addition, it provides a well-documented help for quick reference.

2.1 PRODUCT DEFINITION

'creagx' is a 16-bit operating system. It coins the basics of all operating systems with innovative ideas and utilization. Each phase of the product is developed keeping an eye on enhancing the processor architecture usability. The security features are also implemented up to a desired extend. 'creagx' is not meant for competing with the existing systems; it just points out some common limitations.

The structure of the operating system is very important in this context. A diagram showing the complete structure of the operating system is given.

THE CREAGX STRUCTURE


COMMAND INTERPRETER-COMMANDS-
DISKVIEWER-EDITOR-HELP-ABOUT-
STORAGE-FILE RECORD STORAGE-FILE
FILESYSTEM-FILE RECORDS-
MBR-SYS-DEVICE DRIVERS-USERMANAGER-KEYWORDS
CHS-79,0,1-18
CHS-78,0,1-18
CHS-1-77,0,1-18
CHS-0,1,1-18
CHS-0,0,1-18

Each sector and track is expected to have a certain set of codings or programs. The values vary for a given head, track and sector. The contents and the specifications are tabulated for easy understanding. A clear idea regarding the working of the operating system can be obtained from this table.

TABLE SHOWING creagx STRUCTURE

CYLINDER NUMBER	HEAD NUMBER	SECTOR NUMBER	CONTENT
0	0	1	BOOT RECORD & LOADER
0	0	2	KEYBOARD ISR(INT 20h)
0	0	3	PRINTER ISR(INT 17h)
0	0	5	COM PORT(INT 14h)
0	0	8	DISK DRIVER
0	0	A	USER VERIFICATION PGM.
0	0	B	USER/ PASSWORD RECORDS
0	1	1	FAT DESCRIPTION
0	1	2	FILE RECORDS
0	1	3	FILE RECORDS
0	1	4	...
.			
.			
.			
77	1	1-18	FILE STORAGE
78	1	1-18	SOURCE CODE
78	0	1-18	HELP AND PSN
79	1	1-18	COMMANDS
79	0	1-18	COMMAND INTERPRETER

2.2 PROJECT PLAN

Booting, as expected is the preliminary phase of operating system execution. As per the previous diagram, the tracks and sectors are allocated. The boot signature aa55 at 07DFEh is preserved without use in any execution, adding to the safety of the MBR file. The loader program loads the MBR initially and from there MBR starts its execution. The MBR will first check the FDD for the 'creagx' file and on obtaining it the booting is triggered.

'creagx' supports multi-user facilities along with security measures like user name and password. The details regarding the user will be stored for future references. Attributes are provided to each file. The user can thus access the permitted file. Avery simple format is used for providing permissions.

The storage of files in 'creagx' is unique. FAT entries are stored from inner to outer track. File storage is done in the reversed direction. To have a complete command over the files, a sophisticated set of commands is implemented. Whatever the command is, the execution is completely based on the permissions allowed.

'creagx' also contains a detailed set of device drivers. All of them are written in assembly code. These device drivers enhance the usability of the operating system. The drivers include the programs for timer, keyboard, monitor, hard disk, floppy drive etc. any references to these devices while using creagx operating system will call these procedures written for device drivers.

'creagx' is also rich in utilities. A utility specifically designed for detecting the Processor Serial Number is added. The PSN is a 96-bit number provided to each Pentium III processor. A wise utilization of this utility can reduce software piracy.

A completely formulated help is the facility provided for quick references for the end user. All the commands are explained in the help. The help also provides the complete documentation of the project thereby joining hands with the users for further enhancement.

❖ SOFTWARE REQUIREMENT SPECIFICATION

3. SOFTWARE REQUIREMENT SPECIFICATION

Software Requirement Specification aims at providing the technical specification of requirements of a project in a consistent and unambiguous manner. As 'creagx' is developed from the basic steps, it does not require the existence of any supporting software except the assembler NASM. The operating system is completely developed in Netwide Assembler and hence the specification of NASM is worth mentioning.

3.1 PRODUCT OVERVIEW

The operating system starts by executing the master boot record. The first creagx file will search the Master Boot Record and loads it in the first sector. Then the booting proceeds and the user is welcomed into the operating system with a user name and password authentication.

After booting, the operating system halts for the user interaction by displaying the prompt. Now the user can perform any operation he wants and he can interact with the operating system with a set of commands. The commands are framed in simple words for easy recollection. Any verification of the command can be viewed referring to the help file.

'creagx' operating system adds attractive features to the basics of the existing operating systems. the unique booting procedure and file system makes the product distinct in nature. The coding used for the various phases of the product can be found in the appendix provided. Utilities like PSN retrieval, editors etc. are implemented to standardize the operating system up to a desired extend. The common peripheral devices are also recognized by our operating system accounting for stress free working.

3.2 FUNCTIONAL REQUIREMENTS

The requirements for each phase are described below.

PHASE	PURPOSE	SOFTWARE REQUIREMENTS
BOOTING	Loads loader in the sector2.	NASM
FILE SYSTEM	i) Maintains FAT entries. ii) File storage and retrieval.	NASM, DDD
DEVICE DRIVERS	i) Interrupt Service Routines. ii) Interfaces peripherals.	NASM
COMMAND INTERPRETER	i) Interprets commands. ii) Executes commands.	NASM, DDD
COMMAND SET	i) Gets the user inputs. ii) Processes Commands. iii) Provides output.	NASM
UTILITIES	Supporting programs for 'creagx'	NASM
HELP	i) About 'creagx' and PSN ii) Source code details	DDD

NASM-IDE:

NASM-IDE is a DOS based system providing a front-end to the Netwide Assembler (NASM). It is developed by Mr.Rob Anderton. NASM-IDE has been designed to provide an interface which should be as easy to use as possible for beginners and experts alike, especially those who are familiar with Borland development products.

Features of NASM-IDE 2.0 include:

- protected mode operation using FPK-Pascal and FreeVision
- further enhancements to the syntax highlighting editor
- keyboard macros
- user defined tools
- large file editing

The NASM is used in all the modules and hence it is the most important tool utilized in our product.

DDD:

DDD stands for Dobaish Disk Doctor. This utility proved to be very helpful in checking the disk storage. The utility provides option to view the sector details of the hard disks and the floppy disks available to the system. We can also change the entries of the boot record using this utility. In the left, we have the offset address. To the next, we have the hex codes of the record. The ASCII equivalents are displayed in the right most part of the utility. The details of the track, head and sector are displayed at the bottom.

Partition tables can be viewed by using this utility.In addition, facilities are provided for saving the altered sector details.

❖ DESIGN DOCUMENT

4. DESIGN DOCUMENT

The development of the 'creagx' operating system is partitioned into a set of modules, which are mutually dependent. The different modules in the course of the project are:

i. Booting

ii. File System

iii. Device Drivers

iv. Command Interpreter

v. Command Set

vi. Utilities

vii. Help

*i) **Booting:***

'creagx' operating system has a very unique booting procedure. As in the case of any other operating system, the master boot record is of most significance in this operating system. The Master Boot Record is the sector at cylinder 0, head 0, sector 1 of a hard disk. At the completion of Power On Self Test (POST), INT19 is called. Usually INT 19 tries to read a boot sector from the first floppy drive. If a boot sector is found on the floppy disk then that boot sector is read into memory at location 0000:7C00 and INT 19 jumps to memory location 0000:7C00.

The small program in the boot sector must locate the first part of the operating system's kernel loader program and read that into memory.

Summary of booting:

1) Loads Sector 1, Track 0, Head 0 of the boot drive (A or C) to absolute address 07C00h-07DFFh

2) Checks the 16-bit word at absolute address 07DFEh for AA55h. This is the boot signature and is used by the BIOS to ensure that the sector contains a value boot sector.

3) If this signature is not present, the BIOS will display a message like "Operating System Not Found"

4) Loads DL with 00h if the boot sector was loaded from drive A,80h if the boot sector was loaded from drive C This way, the boot sector can determine which drive it was booted from.

5) Jumps to 0000:7C00h, which is the start of the boot sector

6) Load the creagx first file of the operating system.

P. bbC

Booting also initializes all the device drivers and updates the interrupt vector table. This module is developed using Netwide assembler. The signature bit is always protected from access and corruption. Hence the preliminary process of booting works smoothly always in this operating system.

ii) File System:

The file system is a simple linked list. The entries in the FAT are tabulated below.

FAT ENTRY	SPECIFICATIONS
FILE NAME	The file name can extend up to 33 characters.
ATTRIBUTES	Attributes contain the following options. | A.F | H.F | S.F | I.F | E.O | W.O | R.O | A.F - ASCII FILE H.F - HIDDEN FILE S.F - SYSTEM FILE I.F - INACCESSIBLE FILE E.O - EXECUTE ONLY W.O - WRITE ONLY R.O - READ ONLY Last bit reserved for future use.
START TRACK,HEAD AND SECTOR	Contains the starting track, head and sector.
END TRACK,HEAD AND SECTOR	Contains the ending track, head and sector.
SIZE OF FILE	Specifies the file size.
DATE OF CREATION	Specifies the date of creation of the file or directory.
DATE LAST USED	Specifies the date on which the file or directory was previously accessed.
DATE LAST MODIFIED	Specifies the date on which the file or directory is modified.

As it is already stated, the file system is different in this operating system compared to the usual structure we observe in the prevailing ones. The FAT entry is stored from inner to outer track. To avoid the relocation of files stored on the addition of each FAT entry.

the file storage starts from the outer most tracks to the inner track. When the difference between the tracks is reduced to one, the user will be warned on the low disk space and the file is neglected. This avoids the stealing of CPU cycles for file relocation after each FAT entry.

iii) Device Drivers:

Device drivers are program that links the various peripherals to the system.The purpose of device drivers is to interface all the common devices to the operating system with all the extended services. It allocates the resources for all these devices. The device drivers are triggered by interrupts, procedures or programs.

'creagx' operating system implements the following device drivers.

Keyboard :

This device driver interfaces keyboards. It is implemented as software interrupt 20h. Any keyboard service is satisfied using this device driver. Sensitive keys like ALT, Control, Left and Right Shifts, Insert, Caps are all displayed on use to enhance usability.

COM port:

• COM port is implemented as an Interrupt Service Routine. It provides the modem status as well as the port status. The familiar interrupt 14h is used for the execution of this device driver program. It also detects a modem connected.

Disk:

This device driver is executed as a separate program. This program includes provisions to display the drive details. A special facility is implemented in this device driver. Provisions to detect whether the current disk is changed or not are provided. Although this device driver is designed for floppy disk drive, errors regarding the hard disks are also detected and displayed.

Printer:

This device driver shows the status of the printer. All sorts of errors regarding the printer like paper out, printer off line etc., are identified and displayed. The interrupt used for printer is INT 17h.

iv) Command Interpreter:

The commands are typed after the display of the prompt. Each command has a key word for its recognition which is explained in the next module. The user typed in words are checked for these keywords and it is checked whether it is directed to any command. In effect, the initial function of the command interpreter is to recognize the key words of the commands. As the keywords are detected, the next step is the location of this command on the disk.

As the command is located, the program for the execution of the command is loaded. The memory allocation and other resource allocations are performed by this module. After loading the program in memory, the subsequent step is the execution of the loaded program. The program does the specified task for the user. But, the work of the command interpreter is not yet over. It has to return the control back to the program that triggers the command, i.e, the interpreter should gain the control back.

As the operating system is not supporting GUI, there should be some provision to distinguish between the line that is already having been processed and the line that the interpreter is currently processing. With reference to the traffic signals, we have represented the current line under processing with a green colour with a blinking cursor. The line of command after execution is represented using the red colour. This means that as the current command completes its execution, the letterings will change to red colour and immediately a new blinking, green cursor appears in the next line.

v) Command Set:

The user can communicate with the system using the command interpreter that provides various commands under the following category:

- **File commands:**

  create

  This command is used to create files. As the user types in this command, the user will be asked to give a file name, along with the attributes. The attributes will also be displayed in order so that the user can type in 1 or 0 for these attributes. The attributes are, ASCII file, Hidden file, System file, Inaccessible file, Execute only, Write only and Read only. A value of 1 indicates that the attribute is satisfied.

delete

Delete is performed to erase a file from the disk. For this purpose both the file name and the user name are compared. The security of a file is ensured by the fact that a different user cannot delete your file.

view

This command is used to view the file record and the data. To enhance security, this command is also executed after checking the user name and the file name.

print

This command is used to print the specified file of the logged in user.

- **User commands:**

list

This command is used to list the files of the current user. Rather than specifying the file name only,the FAT entries of the specified file is also displayed on execution of this command.

encrypt

This command encrypts the data of a given file of the current user. The encryption key is obtained by referring to the memory based on the user name we are logging in. this means that the encryption key is unique.

decrypt

Decryption of an encrypted file of the logged in user is performed by this command. This command also refers the memory for the decryption key. This key is also unique in nature.

- **Disk commands**

  capacity

This command specifies the size of the floppy drive. It is capable of recognizing all types of floppy drives.

- **Device commands**

  stat

This gives the current status of the floppy disk. It detects the read and write errors for the operation last performed.

vi) Utilities:

'creagx' provides two basic utilities. They are the disk viewer and the editor.

- **Disk Viewer:**

The disk viewer is the utility by which we can view the details of the floppy disk sector by sector. The head number, track number and sector number are displayed for quick reference. On reaching the formatted part of the floppy, the division symbol is displayed. The division symbol is having an ASCII value of F6. We can scroll between the sectors by using the arrow keys.

- **Editor:**

The editor we have implemented is like any other editor we came across in any assembler. We can change the text details using this editor. Any modifications of the data as well as any text file can be saved and the information can be viewed using the disk viewer utility.

vii) Help:

A detailed help file is provided for all the commands in this operating system. Any references regarding the commands can be satisfied using this help file. Basically, the help of 'creagx' operating system is classified into three.

- About 'creagx'
- About PSN
- Complete Source Code

Also, the help file contains information regarding shortcut keys. Any sort of a technical query regarding the operating system will also be managed by the help. The source code is kept open welcoming any enhancements. Each and every module is explained with sufficient documentation. The details about the Pentium III PSN architecture is also provided to the user enabling its wise use in networking.

PRODUCT TESTING

5. PRODUCT TESTING

Any product before completion needs to be tested for the purpose of enhancing reliability. Testing, is defined as a measure to reduce risks and loss associated to an acceptable level. This product, being an operating system, the need for testing and debugging is optimum. A variety of testing techniques and strategies are developed for the software products. But each product should follow the principle " ***Too little testing is a crime; too much testing is a sin***".

5.1 OBJECTIVES OF TESTING:

1) Investigate structural properties of source code.
2) Exercise the code with nominal inputs.
3) Determine the execution time of each unit of the product.
4) Determine the breaking point of the product.
5) Check the program throughput, response time and device utilization.
6) Determine the optimum traverse path for execution.
7) Detect missing paths, computational and domain errors.
8) Check the validity at each stage of development of the product.
9) Check the overall validity of final products against needs and requirements

2.2 UNIT AND INTEGRATED TESTING:

Among the wide range of testing strategies available, unit and integration testing are the most prevailing.

The basic structure of testing is as follows:

Coding and Debugging → Unit testing → Integration testing

Unit testing comprises the set of tests performed by an individual programmer prior to the integration of the unit into a large system. Unit testing includes the application of functional tests, performance tests, stress tests and finally structure tests.

Integration testing includes bottom up, top down and sandwitch strategies. As far as 'creagx' operating system is concerned, the bottom up strategy is adopted. Each unit of the operating system is tested, then each subsystem is tested and finally the complete product is exposed to testing.

5.2 TEST RESULTS :

The test results for the tests performed for each module is described below.

i) Booting:

On booting, by executing the CPUID instruction, the PSN is tested. The loading of the Interrupt Service Routines and the updation of the Interrupt Vector Table needs to be verified. The verification is performed using the Dobaish Disk Doctor.

ii) File System:

The initial phase of testing needs to be focussed on searching of the specified files by the user. Again testing is to be performed whether the module is integrated with the command system completely. The checking is also performed to ensure whether the file storage is performed precisely.

iii) Device Drivers:

The first criteria for checking is that whether the error messages are displayed for any errors on the flopy disk. Testing is also performed to verify the status of the COM por as well as modem.

iv) Command Interpreter:

The command needs to be identified properly. This is the basic area that needs to be tested in a command interpreter. Testing is also performed to ensure the correct display style of the prompt.

v) Command Set:

The integration of the command set with the file system is worth testing. Also the verification is performed to know whether the correct search routine is employed. Another matter of concern is whether the called command provides the specific purpose.

vi) Utilities:

The disk viewer is checked to verify its capacity to display the complete sectors. The updation process of the editor is also verified using the disk viewer. Tests are performed to check its integration with the file search routine inorder to account for the reliability of the utility.

v) Help:

A verification is performed to know whether the help covers the complete operating system features.

❖ FUTURE ENHANCEMENTS

6. FUTURE ENHANCEMENTS

'creagx' currently is framed in a very basic implementation model. It, at present, does not support Graphical User Interface. As the need of upgrading user facilities goes on increasing day by day, 'creagx' needs to have a GUI version developed in the near future.

This 'creagx'operating system is a 16-bit operating system. The current utilities and applications require atleast a 32-bit operating system for smooth execution. Hence, the upgradation of 'creagx' to 32-bit is another area of serious concern.

As the current version of this operating system works on the floppy disk. the next version of this operating system needs to be designed in such a way that it adapts itself to hard disks.

The 'creagx' operating system specifies the Processor Serial Number (PSN) utility. Since this facility can be used to distinguish between individual systems, online checking for detecting software piracy can be performed on developing corresponding applications. 'creagx' operating system would like to have such an application developed under it.

As the 'creagx' operating system likes to grow to new heights, the suggestions from the users need to be given sufficient importance. For the same reason the complete source code of the operating system is provided in the help. This open source code might thereby trigger the development of a new, better, sophisticated version of 'creagx' operating system.

CONCLUSION

7. CONCLUSION

'creagx' operating system is a joint venture to highlight those aspects of operating systems which the prevailing systems failed to represent. The implementation of each module is done after thorough testing to avoid malfunctioning.

As we have already specified, 'creagx' operating system is not designed for competing with the existing systems. It is a blend of unique characteristics with the basics of operating systems. The highlight of 'creagx' operating system is the recognition of PSN architecture of the Pentium III processor. Although the architecture is not present in the latest Pentium IV processor, we expect 'creagx' to provide the inspiration to bring that architecture back to existence.

This page is a termination only for the premier version of 'creagx' operating system. It certainly does not put a full stop to the enhancement works undergoing to affix the product to a more sophisticated software world.

BIBLIOGRAPHY

BIBLIOGRAPHY

- OPERATING SYSTEM DESIGN AND IMPLEMENTATION
  - ANDREW S. TANENBAUM.

- ADVANCED MICROPROCESSORS AND IBM-PC ASSEMBLY LANGUAGE PROGRAMMING
  - K. UDAYAKUMAR AND B. S. UMASHANKER.

- DOS 5. A DEVELOPER GUIDE
  - AL WILLIAMS.

- BIOS INTERRUPTS USING C/C++
  -TAYLOR BILLY.

- PENTIUM III INSTRUCTION SET AND USER MANUAL
  -INTEL CORPORATIONS.

- OPERATING SYSTEM CONCEPTS
  -H.M DEITEL.

APPENDIX

APPENDIX

SOURCE CODE

```
;THIS PROGRAM IS THE MASTER BOOT RECORD FOR CREAGX OPERATING
SYSTEM
[SECTION .text]              ; this is the code segment
%macro setcurpos 2           ;macro to set cursor
      mov ah,2
      mov bh,0
      mov dh,%1
      mov dl,%2
      int 10h
%endmacro
%macro   clrscr 0           ;macro to clear screen
      mov cx,2000
      label mov ah,14
      mov al,' '
      xor bh,bh
      int 10h
      loop label
      setcurpos 0,0
%endmacro
start:                       ;start of main program
clrscr
mov ax,100h
mov es,ax
mov di,0
mov ah,2
mov al,2
mov ch,0
mov cl,2
mov dh,0
mov dl,0
mov bx,0
int 13h
;************************initialise ivt
mov ax,0000
mov es,ax
mov di,128
mov BYTE [es:di],00
inc di
mov BYTE [es:di],00
inc di
mov BYTE [es:di],00
```

```
inc di
mov BYTE [es:di],01h
;*************************************
mov ax,200h
mov es,ax
mov di,0
mov ah,2
mov al,1
mov ch,0
mov cl,4
mov dh,0
mov dl,0
mov bx,0
int 13h
;************************initialise ivt
mov ax,0000
mov es,ax
mov di,92
mov BYTE [es:di],00
inc di
mov BYTE [es:di],00
inc di
mov BYTE [es:di],00
inc di
mov BYTE [es:di],02h
;****************************
mov ax,300h
mov es,ax
mov di,0
mov ah,2
mov al,2
mov ch,0
mov cl,5
mov dh,0
mov dl,0
mov bx,0
int 13h
;************************initialise ivt
mov ax,0000
mov es,ax
mov di,80
mov BYTE [es:di],00
inc di
mov BYTE [es:di],00
inc di
mov BYTE [es:di],00
```

```
inc di
mov BYTE [es:di],03h
;**************************************************
mov ax,400h
mov es,ax
mov di,0
mov ah,2
mov al,3
mov ch,0
mov cl,7
mov dh,0
mov dl,0
mov bx,0
int 13h
;*************************initialise ivt
mov ax,0000
mov es,ax
mov di,76
mov BYTE [es:di],00
inc di
mov BYTE [es:di],00
inc di
mov BYTE [es:di],00
inc di
mov BYTE [es:di],04h
;*******************************************************
mov ax,1000h
mov es,ax
mov bx,0
mov ah,2
mov al,1
mov ch,0
mov cl,10
mov dh,0
mov dl,0
int 13h

mov ax,1000h
mov es,ax
mov ds,ax
mov ss,ax
mov di,0
jmp 1000h:0000
;**********************************************
times 510-($-$$) db 0
dw 0xaa55
```

```
;THIS PROGRAM DISPLAYS THE PROCESSOR SERIAL NUMBER
[BITS 32]           ;this program uses the 32-bit addressing mode
[SECTION .text]     ;this is the code segment
%macro cpuid 0      ;this is the macro to define the cpuid
      db 0x0f,0xa2
%endmacro
%macro dispchar 0  ;this is the macro to display the character
      mov ah,14
      mov bh,0
      int 10h
%endmacro
proc:                    ;this procedure is to convert the bin to ascii
      push ax
      shr al,4
      cmp al,9
      jle near ndeci
      sub al,9
      add al,64
      dispchar
      jmp near next
ndeci   add al,48
      dispchar
next    pop ax
      and al,15
      cmp al,9
      jle near deci
      sub al,9
      add al,64
      dispchar
      jmp near pover
```

```
deci    add al,48
        dispchar
pover   ret


start:                          ; this is the start of the program
        mov ax,1010h
        mov es,ax
        mov ds,ax
        mov di,0
        mov eax,1
        cpuid
        mov DWORD [es:di],eax
        inc di
        mov eax,3
        cpuid
        mov DWORD [es:di],edx
        inc di
        mov DWORD [es:di],ecx
        mov di,0
        mov ax,1010h
        mov es,ax
        mov cx,12
        disppsn
        mov BYTE al,[es:di]
        call proc
        inc di
        loop  disppsn
```

THIS PROGRAM IS THE KEYBOARD DRIVER FOR CREAGX OPERATING SYSTEM

```
[BITS 16]                ; 16 BIT ADDRESSING MODE
[ORG 0X0000]             ; ADDRESS OF THE PROGRAM IN RAM
;------------------------------------------------------------
[SECTION  .text]
```

```
%macro  dispnum 0
      cmp al,47
      jng num1
      cmp al,58
      jnl num1
      push ax
      mov ah,14
      xor bh,bh
      int 10h
      pop ax
      num1 nop
%endmacro

%macro dispsletter 0
      cmp al,96
      jng letter
      cmp al,123
      jnl letter
      push ax
      mov ah,14
      xor bh,bh
      int 10h
      pop ax
      letter nop
%endmacro

%macro dispcletter  0
      cmp al,64
      jng letter1
      cmp al,91
      jnl letter1
      push ax
      mov ah,14
      xor bh,bh
      int 10h
      pop ax
      letter1 nop
%endmacro

%macro otherkeys 0
      cmp ah,1
      je near exit
      cmp ah,77
      jne  near lbl1
      inccursor
  lbl1  cmp ah,75
```

```
      jne  near lbl2
      deccursor
 lbl2  cmp ah,14
      jne near lbl3
      deccursor
      erasechar
      deccursor
 lbl3  cmp ah,83
      jne near lbl4
      erasechar
      deccursor
 lbl4  cmp ah,28
      jne near lbl5
      incrow
 lbl5  cmp ah,71
      jne near lbl6
      mov ah,3
      mov bh,0
      int 10h
      mov dl,7
      mov ah,2
      mov bh,0
      int 10h
 lbl6   cmp ah,57
      jne near lbl7
      mov ah,14
      mov bh,0
      int 10h
 lbl7   cmp ah,15
      jne near lblend
      mov ah,3
      mov bh,0
      int 10h
      add dl,8
      mov ah,2
      mov bh,0
      int 10h
 lblend nop

%endmacro

%macro incrow 0
      mov ah,3
      mov bh,0
      int 10h
      inc dh
```

```
     mov dl,7
     mov ah,2
     mov bh,0
     int 10h


%endmacro


%macro erasechar 0
     mov ah,14
     mov al,32
     mov bh,0
     int 10h
%endmacro

%macro inccursor 0
      mov ah,3
      mov bh,0
      int 10h
      inc dl
      mov ah,2
      mov bh,0
      int 10h
%endmacro

%macro deccursor 0
      mov ah,3
      mov bh,0
      int 10h
      dec dl
      mov ah,2
      mov bh,0
      int 10h
%endmacro

%macro writestring 4       ;macro to write the string
   getcursor
   mov bp,%1
   mov ah,13h           ;move function number to ah
   mov al,1            ;move write mode to al
   mov bh,0             ;move vdu page no to  bh
   mov bl,14            ;move attribute to bl
   mov cx,%2            ;move the stringlength to cx as parameter2
   mov dh,%3            ;move row to dh reg
   mov dl,%4            ;move col to dl reg
   int 10h              ;call the interrupt
   setcursor
%endmacro


%macro shiftstatus 0
   getcursor
   mov ah,2
   mov bh,0
   mov dh,24
   mov dl,0
   int 10h
   mov ah,9
   mov al,20h
   mov bh,0
   mov bl,0
   mov cx,80
   int 10h
   setcursor
   mov ah,2
   int 16h
   mov BYTE [shift],al
   and al,1
   cmp al,1
   jne near label1
   writestring rshift,6,24,5
 label1  mov al,[shift]
       and al,2
       cmp al,2
       jne near label2
       writestring lshift,6,24,12
 label2  mov al,[shift]
       and al,4
       cmp al,4
       jne near label3
       writestring ctrl,4,24,18
 label3  mov al,[shift]
       and al,8
       cmp al,8
       jne near label4
       writestring alt,3,24,23
 label4  mov al,[shift]
       and al,16
       cmp al,16
       jne near label5
       writestring scroll,6,24,55
```

```
label5  mov al,[shift]
      and al,32
      cmp al,32
      jne near label6
      writestring num,3,24,34
label6  mov al,[shift]
      and al,64
      cmp al,64
      jne near label7
      writestring caps,4,24,39
label7  mov al,[shift]
      and al,128
      cmp al,128
      jne near labelend
      writestring insert,6,24,45
labelend nop
%endmacro

%macro getcursor 0
      mov ah,3
      mov bh,0
      int 10h
      mov BYTE [row],dh
      mov BYTE [col],dl
%endmacro

%macro setcursor 0
      mov ah,2
      mov bh,0
      mov dh,[row]
      mov dl,[col]
      int 10h
%endmacro

start  pusha
      mov ax,100h
      mov es,ax
      mov ds,ax
      xor ah,ah
      int 16h
      dispnum
      dispsletter
      dispcletter
      otherkeys
      shiftstatus
exit   popa
```

```
      iret
[SECTION .data]
rshift db 'RSHIFT'
lshift db 'LSHIFT'
ctrl   db 'CTRL'
alt    db 'ALT'
scroll db 'SCROLL'
num    db 'NUM'
caps   db 'CAPS'
insert db 'INSERT'

[SECTION .bss]
shift resb 1
row   resb 1
col   resb 1



;THIS PROGRAM IS THE DISK DRIVER FOR THE CREAGX OPERATING
SYSTEM
;------------------------------------------------------------------
[ORG 0X0000]
[SECTION  .text]
   jmp near start         ;code starts from label start
%macro writestring 4      ;macro to write the string
   mov ax,%1
   mov bp,ax
   mov ah,13h             ;move function number to ah
   mov al,1               ;move write mode to al
   mov bh,0               ;move vdu page no to  bh
   mov bl,14              ;move attribute to bl
   mov cx,%2              ;move the stringlength to cx as parameter2
   mov dh,%3              ;move row to dh reg
   mov dl,%4              ;move col to dl reg
   int 10h                 ;call the interrupt
%endmacro

start                     ;start of code for exection
  mov BYTE [stat],ah
 pusha
  mov ax,400h
  mov ds,ax
  mov es,ax

  mov BYTE ah,[stat]
  cmp ah,0               ;compare the value of ah against the error code
  je near err0
```

```
cmp ah,1
je near err1

cmp ah,2
je near err2

cmp ah,3
je near err3

cmp ah,4
je near err4

cmp ah,5
je near err5

cmp ah,6
je near err6

cmp ah,7
je near err7

cmp ah,8
je near err8

cmp ah,9
je near err9

cmp ah,10
je near erra

cmp ah,11
je near errb

cmp ah,12
je near errc

cmp ah,13
je near errd

cmp ah,14
je near erre

cmp ah,15
je near errf
```

```
cmp ah,16
je near err10

cmp ah,17
je near err11

cmp ah,32
je near err20

cmp ah,64
je near err40

cmp ah,128
je near err80

cmp ah,170
je near erraa

cmp ah,187
je near errbb

cmp ah,204
je near errcc

cmp ah,224
je near erre0

cmp ah,255
je near errff
jmp near exit

err0 mov si, error0                    ;display the error message
writestring error0,8,1,0
jmp near exit

err1 mov si, error1
writestring error1,24,2,0
jmp near exit

err2 mov si, error2
writestring error2,22,3,0
jmp near exit

err3 mov si, error3
writestring error3,19,4,0
jmp near exit
```

```
err4 mov si, error4
writestring error4,16,5,0
jmp near exit

err5 mov si, error5
writestring error5,12,6,0
jmp near exit

err6 mov si, error6
writestring error6,19,7,0
jmp near exit

err7 mov si, error7
writestring error7,19,8,0
jmp near exit

err8 mov si, error8
writestring error8,11,9,0
jmp near exit

err9 mov si, error9
writestring error9,17,10,0
jmp near exit

erra mov si, errora
writestring errora,15,11,0
jmp near exit

errb mov si, errorb
writestring errorb,14,12,0
jmp near exit

errc  mov si, errorc
writestring errorc,20,13,0
jmp near exit

errd mov si, errord
writestring errord,35,14,0
jmp near exit

erre mov si, errore
writestring errore,34,15,0
jmp near exit

errf  mov si, errorf
```

```
writestring errorf,34,16,0
jmp near exit

err10  mov si, error10
writestring error10,35,17,0
jmp near exit

err11 mov si, error11
writestring error11,24,18,0
jmp near exit

err20  mov si, error20
writestring error20,17,19,0
jmp near exit

err40 mov si, error40
writestring error40,21,20,0
jmp near exit

err80  mov si, error80
writestring error80,23,21,0
jmp near exit

erraa   mov si, erroraa
writestring erroraa,15,22,0
jmp near exit

errbb mov si, errorbb
writestring errorbb,15,23,0
jmp near exit

errcc mov si, errorcc
writestring errorcc,11,24,0
jmp near exit

erre0 mov si, errore0
writestring errore0,20,1,30
jmp near exit

errff mov si, errorff
writestring errorff,22,2,30

exit popa
   iret

[SECTION .data]
```

```
error0  db 'no error$'
error1  db 'invalid function request$'
error2  db 'address mark not found$'
error3  db 'diskwrite protected$'
error4  db 'sector not found$'
error5  db 'reset failed$'
error6  db 'floppy disk removed$'
error7  db 'bad parameter table$'
error8  db 'DMA failure$'
error9  db 'DMA crossed 64 kb$'
errora  db 'bad sector flag$'
errorb  db 'bad track flag$'
errorc  db 'media type not found$'
errord  db 'invalid number of sectors on format$'
errore  db 'control data address mark detected$'
errorf  db 'DMA arbitration level out of range$'
error10 db 'uncorrectable CRC or ECC data error$'
error11 db 'ECC corrected data error$'
error20 db 'controller failed$'
error40 db 'seek operation failed$'
error80 db 'drive failed to respond$'
erroraa db 'drive not ready$'
errorbb db 'undefined error$'

errorcc db 'write fault$'
errore0 db 'status register error$'
errorff db 'sense operation failed$'
[section .bss]
stat resb 1



;THIS PROGRAM GIVES ABOUT THE CHANGE IN DISK
[BITS 16]              ; 16 BIT ADDRESSING MODE
[ORG 0X0100]            ; ADDRESS OF THE PROGRAM IN RAM
;------------------------------------------------------------
[SECTION  .text]
%macro writestring 4      ;macro to write the string
   mov ax,%1            ;move the offset value to bp thru ax
   mov bp,ax
   mov ah,13h           ;move function number to ah
   mov al,1            ;move write mode to al
   mov bh,0             ;move vdu page no to  bh
   mov bl,14            ;move attribute to bl
   mov cx,%2             ;move the stringlength to cx as parameter2
   mov dh,%3             ;move row to dh reg
   mov dl,%4            ;move col to dl reg
   int 10h            ;call the interrupt
%endmacro

   mov ah,16h
   mov dl,0
   int 13h

   cmp ah,0
   je near mess
   jmp nmess



mess writestring message,31,10,10
    jmp exit
nmess writestring nmessage,33,11,10
exit xor ah,ah
     int 16h
     mov ax,$4c
     int 21h

[SECTION .data]
message db 'THE DISK IN DRIVE IS NOT CHANGED'
nmessage db 'THE DISK IN THE DRIVE IS CHANGED'


; THIS IS THE SAMPLE PROGRAM TO READ THE DISK
[ORG 0X0100]
[BITS 16]
[SECTION .text]
;---------------------
mov ax,1000h
mov es,ax
mov bx,0
mov ah,2
mov al,1
mov ch,0
mov cl,1
mov dh,0
mov dl,0
int 13h
;---------------------
    mov di,0
    mov ah,14
    mov bh,0
    mov cx,512
```

```
  label     mov BYTE al,[es:di]
          inc di
          int 10h
          loop  label

;--------------------------------
xor ah,ah
int 16h
;--------------------------------

;THIS PROGRAM REPORTS THE DRIVE TYPE
[BITS 16]              ; 16 BIT ADDRESSING MODE
[ORG 0X0100]              ; ADDRESS OF THE PROGRAM IN RAM
;--------------------------------------------------------------
[SECTION  .text]
%macro writestring 4      ;macro to write the string
   mov ax,%1             ;move the offset value to bp thru ax
   mov bp,ax
   mov ah,13h             ;move function number to ah
   mov al,1             ;move write mode to al
   mov bh,0              ;move vdu page no to  bh
   mov bl,14             ;move attribute to bl
   mov cx,%2              ;move the stringlength to cx as parameter2
   mov dh,%3              ;move row to dh reg
   mov dl,%4             ;move col to dl reg
   int 10h             ;call the interrupt
%endmacro
   push es
   mov ah,8
   mov dl,0
   int 13h

   cmp bl,1
   je  near disk1
   cmp bl,2
   je  near disk2
   cmp bl,3
   je  near disk3
   cmp bl,4
   je near disk4
   jmp exit


 disk1 pop es
 writestring mess,19,10,10
```

```
    writestring dtype1,20,11,10
    jmp exit
disk2 pop es
writestring mess,19,10,10
    writestring dtype2,20,11,10
    jmp exit

disk3 pop es
writestring mess,19,10,10
    writestring dtype3,20,11,10
    jmp exit

disk4 pop es
writestring mess,19,10,10
    writestring dtype4,20,11,10

exit xor ah,ah
   int 16h
   mov ax,$4c
   int 21h

[SECTION .data]
mess   db 'THE DRIVE TYPE IS: '
dtype1 db '360kb ,40 track,5.25 inch'
dtype2 db '1.2MB ,80 track,5.25 inch'
dtype3 db '720kb ,80 track,3.5  inch'
dtype4 db '1.44MB,80 track,3.5  inch'

;THIS PROGRAM IS THE PRINTER DRIVER FOR CREAGX OPERATING
SYSTEM
[BITS 16]              ; 16 BIT ADDRESSING MODE
;[ORG 0X0100]              ; ADDRESS OF THE PROGRAM IN RAM
;--------------------------------------------------------------
[SECTION  .text]

%macro writestring 4      ;macro to write the string
   mov ax,%1            ;move the offset value to bp thru ax
   mov bp,ax
   mov ah,13h           ;move function number to ah
   mov al,1            ;move write mode to al
   mov bh,0             ;move vdu page no to  bh
   mov bl,14            ;move attribute to bl
   mov cx,%2             ;move the stringlength to cx as parameter2
   mov dh,%3             ;move row to dh reg
   mov dl,%4            ;move col to dl reg
   int 10h            ;call the interrupt
```

```
%endmacro


   mov BYTE [shift],ah

   pusha
   mov ax,200h
   mov es,ax
   mov ds,ax

   mov BYTE ah,[shift]
   and ah,1
   cmp ah,1
   jne near label1
   writestring err1,17,17,20
label1  mov ah,[shift]
      and ah,8
      cmp ah,8
      jne near label2
      writestring err2,24,18,20
label2  mov ah,[shift]
      and ah,16
      cmp ah,16
      jne near label3
      writestring err3,16,19,20
      label3  mov ah,[shift]
      and ah,32
      cmp ah,32
      jne near label4
      writestring err4,9,20,20
label4  mov ah,[shift]
      and ah,64
      cmp ah,64
      jne near label5
      writestring err5,19,21,20
label5  mov ah,[shift]
      and ah,128
      cmp ah,128
      jne near labelend
      writestring err6,16,22,20
labelend nop

exit   popa
      iret
[SECTION .data]
err1 db 'printer timed out$'
```

```
err2 db 'i/o error or printer off$'
err3 db 'printer selected$'
err4 db 'paper out$'
err5 db 'printer acknowledge$'
err6 db 'printer not busy$'
[SECTION .bss]
shift resb 1


;THIS PROGRAM IS THE COMMAND INTERPRETER FOR THE CREAGX
[ORG 0X0100]
[BITS 16]
[SECTION .text]
        jmp start
%macro setcurpos 2
        mov ah,2
        mov bh,0
        mov dh,%1
        mov dl,%2
        int 10h
%endmacro
%macro dispchr 0
        mov ah,14
        xor bh,bh
        int 10h
%endmacro
%macro keypress 0
        xor ah,ah
        int 16h
%endmacro
%macro readcreate 0
mov ax,8000h
mov es,ax
mov di,0
xor bx,bx
mov ah,2
mov al,4
mov ch,1
mov cl,1
mov dh,0
mov dl,0
int 13h
mov ax,8000h
mov es,ax
mov ds,ax
call 8000h:0000
```

```
%endmacro
dispchar:
        cmp al,47
        jng num1
        cmp al,58
        jnl num1
        mov ah,14
        xor bh,bh
        int 10h
        num1 ret
displetter:
        cmp al,96
        jng letter
        cmp al,123
        jnl letter
        mov ah,14
        xor bh,bh
        int 10h
        letter ret
dispcaps:
        cmp al,64
        jng letter1
        cmp al,91
        jnl letter1
        mov ah,14
        xor bh,bh
        int 10h
        letter1 ret

%macro   clrscr 0
        mov cx,2000
        label mov ah,14
        mov al,' '
        xor bh,bh
        int 10h
        loop label
        setcurpos 0,0
%endmacro
%macro getcursor 0
        mov ah,3
        xor bh,bh
        int 10h
%endmacro
%macro inccur  0
        mov ah,2
        xor bh,bh
        add dl,1
        int 10h
%endmacro
%macro deccur  0
        mov ah,2
        xor bh,bh
        sub dl,1
        int 10h
%endmacro
%macro exit 0
        mov ah,$4c
        int 21h
%endmacro
%macro incline 0
        mov ah,3
        xor bh,bh
        int 10h
        mov ah,02
        xor bh,bh
        xor dl,dl
        int 10h

        mov si,prompt
        char lodsb
        cmp al,'$'
        je keey
        mov ah,9
        xor bh,bh
        mov bl,4
        mov cx,1
        int 10h
        getcursor
        inccur
        jmp char
        keey hlt
        xor dl,dl
        add dh,1
        mov ah,2
        xor bh,bh
        int 10h
        call wrtprompt
%endmacro

%macro dispdate    0

        setcurpos 24,69
```

```
mov ah,4
int $1a
push dx

and dl,240
rol dl,4
add dl,48
mov al,dl
dispchr

pop dx

and dl,15
add dl,48
mov al,dl
dispchr

mov al,45
dispchr
push dx

and dh,240
rol dh,4
add dh,48
mov al,dh
dispchr

pop dx

and dh,15
add dh,48
mov al,dh
dispchr

mov al,45
dispchr

push cx

and ch,240
rol ch,4
add ch,48
mov al,ch
dispchr

pop cx
```

```
        and ch,15
        add ch,48
        mov al,ch
        dispchr

        push cx

        and cl,240
        rol cl,4
        add cl,48
        mov al,cl

        dispchr

        pop cx

        and cl,15
        add cl,48
        mov al,cl
        dispchr
%endmacro

wrtprompt:
         mov si,prompt
         chr lodsb
         cmp al,'S'
         je key
         mov ah,9
         xor bh,bh
         mov bl,138
         mov cx,1
         int 10h
         getcursor
         inccur
         jmp chr
         key hlt
         ret
;end of procedure

keycheck:
         getcursor
         cmp dl,7
         jg mc
         setcurpos dh,8
         mov al,8
        call dispchar
        mc nop
        ret
;-----------------------------------------------------------------

start
        push ds
        push es
        push si
        push di
        mov ax,1000h
        mov es,ax
        mov di,0
        mov si,create
        mov cx,6

 cr      mov al,[ds:si]
        mov [es:di],al
        inc di
        inc si
        loop cr

        mov si,view
        mov cx,4
 vie     mov al,[ds:si]
        mov [es:di],al
        inc di
        inc si
        loop vie

        mov si,delete
        mov cx,3
 del     mov al,[ds:si]
        mov [es:di],al
        inc di
        inc si
        loop del
        mov si,encrypt
        mov cx,4
 enc      mov al,[ds:si]
        mov [es:di],al
        inc di
        inc si
        loop enc
        mov si,decrypt
        mov cx,4
```

```
decr    mov al,[ds:si]
        mov [es:di],al
        inc di
        inc si
        loop decr
        mov si,attribute
        mov cx,4
attr    mov al,[ds:si]
        mov [es:di],al
        inc di
        inc si
        loop attr
        mov si,print
        mov cx,5
pri     mov al,[ds:si]
        mov [es:di],al
        inc di
        inc si
        loop pri
;------------------------------------------------------------

        mov di,0
        clrscr
        dispdate
        setcurpos 0,0
        call wrtprompt
        mov cx,0

stroke    nop

nextline  mov ax,2000h
          mov es,ax
          keypress
          cmp ah,1
          je near ext
         mov BYTE [es:di],al
         inc di
         inc cx
         call displetter
         call dispchar
         call dispcaps
         cmp al,13
         je near kin
         jmp near stroke

kin
```

```
mov di,0
mov ax,1000h
mov ds,ax
mov si,0
mov cx,6
repe cmpsb
cmp cx,0
je near cc
mov di,0
mov si,6
mov cx,4
repe cmpsb
cmp cx,0
je near equal

mov di,0
mov si,10
mov cx,4
repe cmpsb
cmp cx,0
je near equal

mov di,0
mov si,13
mov cx,4
repe cmpsb
cmp cx,0
je near equal

mov di,0
mov si,17
mov cx,4
repe cmpsb
cmp cx,0
je near equal

mov di,0
mov si,21
mov cx,4
repe cmpsb
cmp cx,0
je near equal

mov di,0
mov si,25
mov cx,5
```

```
        repe cmpsb
        cmp cx,0
        je near equal

;---------------------------------------------------------------
        pop di
        pop si
        pop es
        pop ds
        incline
        push ds
        push es
        push si
        push di
        mov di,0
        mov cx,0
        jmp near nextline
equal   mov al,49
        dispchr
        jmp bye
cc      readcreate
ext     nop
bye     keypress
        exit
        ;retf

[SECTION .data]
prompt  DB 'CREAGX>$'
create  db 'create'
view    db 'view'
delete  db 'del'
encrypt  db 'encr'
decrypt  db 'decr'
attribute db 'attr'
print   db 'print'

[SECTION .bss]
command  resb   20


;THIS PROGRAM IS THE HANDLER FOR SERIAL PORT AND MODEM LINE
;[ORG 0X0000]           ; ADDRESS OF THE PROGRAM IN RAM
;---------------------------------------------------------------
[SECTION  .text]

%macro writestring 4      ;macro to write the string
```

```
   mov ax,%1              ;move the offset value to bp thru ax
   mov bp,ax
   mov ah,13h             ;move function number to ah
   mov al,1               ;move write mode to al
   mov bh,0               ;move vdu page no to  bh
   mov bl,14              ;move attribute to bl
   mov cx,%2              ;move the stringlength to cx as parameter2
   mov dh,%3              ;move row to dh reg
   mov dl,%4              ;move col to dl reg
   int 10h                ;call the interrupt
%endmacro


%macro portstatus   0
   and ah,1
   cmp ah,1
   jne near label1
   mov si,portstat0
   writestring portstat0,13,16,20
label1  mov BYTE ah,[pshift]
      and ah,2
      cmp ah,2
      jne near label2
       mov si,portstat1
      writestring portstat1,22,17,20
label2  mov BYTE ah,[pshift]
      and ah,4
      cmp ah,4
      jne near label3
     mov si,portstat2
     writestring portstat2,21,18,20
label3  mov BYTE ah,[pshift]
      and ah,8
      cmp ah,8
      jne near label4
      mov si,portstat3
      writestring portstat3,22,19,20
label4  mov BYTE ah,[pshift]
      and ah,16
      cmp ah,16
      jne near label5
      mov si,portstat4
      writestring portstat4,14,20,20
label5  mov BYTE ah,[pshift]
      and ah,32
      cmp ah,32
      jne near label6
      mov si,portstat5
      writestring portstat5,31,21,20
label6  mov BYTE ah,[pshift]
      and ah,64
      cmp ah,64
      jne near label7
      mov si,portstat6
      writestring portstat6,29,22,20
label7  mov BYTE  ah,[pshift]
      and ah,128
      cmp ah,128
      jne near labelend
      mov si,portstat7
      writestring portstat7,14,23,20
labelend nop
%endmacro

%macro modemstatus   0
   and al,1
   cmp al,1
   jne near mlabel1
   mov si,modemstat0
   writestring modemstat0,23,1,20

mlabel1  mov BYTE al,[mshift]
      and al,2
      cmp al,2
      jne near mlabel2
      mov si,modemstat1
      writestring modemstat1,24,2,20

mlabel2  mov BYTE al,[mshift]
      and al,4
      cmp al,4
      jne near mlabel3
   mov si,modemstat2
      writestring modemstat2,27,3,20

mlabel3  mov BYTE al,[mshift]
      and al,8
      cmp al,8
      jne near mlabel4
   mov si,modemstat3
      writestring modemstat3,36,4,20
```

```
mlabel4  mov BYTE al,[mshift]
      and al,16
      cmp al,16
      jne near mlabel5
      mov si,modemstat4
      writestring modemstat4,13,5,20

mlabel5  mov BYTE al,[mshift]
      and al,32
      cmp al,32
      jne near mlabel6
      mov si,modemstat5
      writestring modemstat5,14,6,20

mlabel6  mov BYTE al,[mshift]
      and al,64
      cmp al,64
      jne near mlabel7
       mov si,modemstat6
      writestring modemstat6,23,7,20

mlabel7  mov BYTE al,[mshift]
      and al,128
      cmp al,128
      jne near mlabelend
       mov si,modemstat7
      writestring modemstat7,26,8,20
mlabelend nop
%endmacro



    mov BYTE [pshift],ah
    mov BYTE [mshift],al
    pusha

    mov ax,300h
    mov es,ax
    mov ds,ax

    mov ah,[pshift]
    portstatus
    mov al,[mshift]
    modemstatus
exit popa
    iret
```

```
[SECTION .data]
portstat0   db 'data is ready'
portstat1   db 'overrun error detected'
portstat2   db 'parity error detected'
portstat3   db 'framing error detected'
portstat4   db 'break detected'
portstat5   db 'transmit holding register empty'
portstat6   db 'transmit shift register empty'
portstat7   db 'time-out error'

modemstat0  db 'change in clear to send'
modemstat1  db 'change in data set ready'
modemstat2  db 'trailing edge ring detector'
modemstat3  db 'change in receive line signal detect'
modemstat4  db 'clear to send'
modemstat5  db 'data set ready'
modemstat6  db 'ring indicator detected'
modemstat7  db 'receive line signal detect'

[SECTION .bss]
pshift resb 1
mshift resb 1

;THIS IS THE CREATE COMMAND FOR THE CREAGX OPERATING SYSTEM
[BITS 16]              ; 16 BIT ADDRESSING MODE
[ORG 0X0100]             ; ADDRESS OF THE PROGRAM IN RAM
;-----------------------------------------------------------------
[SECTION  .text]
  %macro writestring 4     ;macro to write the string
    mov ax,%1              ;the offset value to bp thru ax
    mov bp,ax
    mov ah,13h             ;move function number to ah
    mov al,1               ;move write mode to al
    mov bh,0               ;move vdu page no to  bh
    mov bl,14              ;move attribute to bl
    mov cx,%2              ;move the stringlength to cx as parameter2
    mov dh,%3              ;move row to dh reg
    mov dl,%4              ;move col to dl
    int 10h                 ;call the interrupt
  %endmacro

  %macro keypress 0      ;MACRO TO GET A KEYPRESS
    xor ah,ah
    int 16h
  %endmacro
  %macro dispchar 0
     mov ah,14
     xor bh,bh
     int 10h
  %endmacro
     push ds
     push es
     push di
     push si

     push ds
     push es
     push di
     push si

     push ds
     push es
     push di
     push si

     push ds
     push es
     push di
     push si
     mov ax,4000h
     mov es,ax
     mov di,0
     mov cx,13
     mov si,username
uname  mov BYTE al,[ds:si]
     mov BYTE [es:di],al
     inc di
     inc si
     loop uname
;----READING DATE FROM SYSTEM---------------------------
      mov ax,250h
      mov es,ax
      mov di,0
      mov ah,4
      int $1a
      push dx
      and dl,240
      rol dl,4
      add dl,48
      mov al,dl
mov BYTE [es:di],al
inc di
dispchar
pop dx
and dl,15
add dl,48
mov al,dl
mov BYTE [es:di],al
inc di
mov BYTE [es:di],'-'
inc di
dispchar
mov al,45
dispchar
push dx
and dh,240
rol dh,4
add dh,48
mov al,dh
mov BYTE [es:di],al
inc di
dispchar
pop dx
and dh,15
add dh,48
mov al,dh
mov BYTE [es:di],al
inc di
mov BYTE [es:di],'-'
inc di
dispchar
mov al,45
dispchar
push cx
and ch,240
rol ch,4
add ch,48
mov al,ch
mov BYTE [es:di],al
inc di
dispchar
pop cx
and ch,15
add ch,48
mov al,ch
mov BYTE [es:di],al
```

```
      inc di
      dispchar
      push cx
      and cl,240
      rol cl,4
      add cl,48
      mov al,cl
      mov BYTE [es:di],al
      inc di
      dispchar
      pop cx
      and cl,15
      add cl,48
      mov al,cl
      mov BYTE [es:di],al

      dispchar
;------GET FILENAME FROM USER---------------------------
      pop si
      pop di
      pop es
      pop ds
      writestring message,34,8,10
      writestring filename,31,10,10
      mov ax,1000h
      mov ds,ax
      mov si,0
      mov cx,0
label1  keypress
      cmp al,13
      je lend
      dispchar
      mov BYTE [ds:si],al
      inc si
      inc cx
      cmp cx,33
      jg lend
      jmp label1
lend    mov ax,si
      mov cx,33
      sub cx,ax
      mov al,' '
fillzero nop
      mov BYTE [ds:si],al
      inc si
      loopnz fillzero
```

```
;---------------FILENAME & USERNAME CHECKING-------
       pop si
       pop di
       pop es
       pop ds
       mov ax,700h
       mov es,ax
       mov di,0
       xor bx,bx
       mov ah,2
       mov al,1
       mov ch,0
       mov cl,1
       mov dh,1
       mov dl,0
       int 13h
       mov di,010ah
       mov BYTE al,[es:di]
       mov BYTE [maxtrack],al
       mov di,010ch
       mov BYTE al,[es:di]
       mov BYTE [maxhead],al
       mov di,010eh
       mov BYTE al,[es:di]
       mov BYTE [maxsect],al
       mov BYTE [curtrack],0
       mov BYTE [curhead],1
       mov BYTE [cursect],2

nextsector
       mov ax,100h
       mov es,ax
       mov bx,0
       mov ah,2
       mov al,1
       mov ch,[curtrack]
       mov cl,[cursect]
       mov dh,[curhead]
       mov dl,0
       int 13h
       push ds
       push es
       push di
       push si
;----------------------------------------
       mov ax,100h
```

```
      mov ds,ax
      mov ax,1000h
      mov es,ax
      mov si,0
      mov di,0
cmpnext       nop
              ;comparing filename
              inc si
              inc si
              mov di,0
              mov cx,33
              repe cmpsb
              cmp cx,0
              jne nequal
              jmp near exit3



nequal  cmp si,445
       jge near npres
       mov ax,si
       mov bl,102
       div bl
       inc al
       mov bl,102
       mul bl
       mov si,ax
       jmp near cmpnext
;-----------------display record-------------
npres
       pop si
       pop di
       pop es
       pop ds
       mov BYTE al,[cursect]
       cmp al,18
       jl near incsect
       mov BYTE [cursect],1
       mov BYTE al,[curhead]
       cmp al,1
       jl near inchead
       mov BYTE [curhead],0
       mov BYTE al,[curtrack]
       inc al
       mov BYTE [curtrack],al
       jmp near next
```

```
inchead mov BYTE [curhead],1
        jmp near next1
incsect mov BYTE al,[cursect]
        inc al
        mov BYTE [cursect],al
next    nop
next1   mov BYTE al,[curtrack]
        mov BYTE ah,[maxtrack]
        cmp al,ah
        jl near nextsector
        mov BYTE al,[curhead]
        mov BYTE ah,[maxhead]
        cmp al,ah
        jl near nextsector
        mov BYTE al,[cursect]

        mov BYTE ah,[maxsect]
        cmp al,ah
        jle near nextsector
;--------------------------------------------------
        pop si
        pop di
        pop es
        pop ds
        mov ax,1000h
        mov ds,ax
        mov si,33
        mov BYTE [ds:si],'#'
        inc si
        writestring permission,41,12,10

        mov BYTE [ds:si],0
        inc si
        writestring ronly,9,13,16
        keypress
        dispchar
        mov BYTE [ds:si],al
        inc si
        writestring wonly,10,14,16
        keypress
        dispchar
        mov BYTE [ds:si],al
        inc si
        writestring xonly,15,15,16
        keypress
        dispchar
```

```
mov BYTE [ds:si],al
inc si
writestring access,12,16,16
keypress
dispchar
mov BYTE [ds:si],al
inc si
writestring system,11,17,16
keypress
dispchar
mov BYTE [ds:si],al
inc si

writestring hidden,11,18,16
keypress
dispchar
mov BYTE [ds:si],al
inc si
writestring ascii,10,19,16
keypress
dispchar
mov BYTE [ds:si],al
inc si
mov si,34
mov BYTE al,[ds:si]
and al,1
sal al,7
mov BYTE [ds:si],al
inc si

mov BYTE al,[ds:si]
and al,1
sal al,6
mov BYTE [ds:si],al
inc si

mov BYTE al,[ds:si]
and al,1
sal al,5
mov BYTE [ds:si],al
inc si

mov BYTE al,[ds:si]
and al,1
sal al,4
mov BYTE [ds:si],al
```

```
      inc si

      mov BYTE al,[ds:si]
      and al,1
      sal al,3
      mov BYTE [ds:si],al
      inc si


      mov BYTE al,[ds:si]
      and al,1
      sal al,2
      mov BYTE [ds:si],al
      inc si

      mov BYTE al,[ds:si]
      and al,1
      sal al,1
      mov BYTE [ds:si],al
      inc si

      mov BYTE al,[ds:si]
      and al,1

      mov BYTE [ds:si],al
      mov si,34
      mov cx,7
      mov BYTE al,[ds:si]
      inc si
addagain nop
      mov BYTE bl,[ds:si]
      inc si
      add al,bl
      loop addagain
      mov si,34
      mov BYTE [ds:si],al
      inc si
      mov BYTE [ds:si],'$'
      mov si,0

;--------------END OF INPUT AREA---------------------------
;----------FORMATION OF FILE RECORD ENTRY------------------
      mov ax,2000h
      mov es,ax
      mov di,0
      mov BYTE [es:di],'#'
```

```
        inc di
        mov BYTE [es:di],'#'
        inc di
        mov si,0
        mov cx,33
writefname
        mov BYTE al,[ds:si]
        mov BYTE [es:di],al
        inc si
        inc di
        loop writefname
        mov BYTE [es:di],'#'
        inc di
        push ds
        push si
        mov ax,4000h
        mov ds,ax
        mov si,0
        mov cx,13
writeuname
        mov BYTE al,[ds:si]
        mov BYTE [es:di],al
        inc si
        inc di
        loop writeuname

        mov BYTE [es:di],'#'
        inc di
        mov ax,250h          ;INITIALIZING DATE SEGMENT
        mov ds,ax
        mov si,0
        mov cx,10
writedate
        mov BYTE al,[ds:si]
        mov BYTE [es:di],al
        inc si
        inc di
        loop writedate
        mov BYTE [es:di],'#'
        inc di
        mov si,0
        mov cx,10
writedate1
        mov BYTE al,[ds:si]
        mov BYTE [es:di],al
        inc si
```

```
	inc di
	loop writedate1
	mov BYTE [es:di],'#'
	inc di
	mov si,0
	mov cx,10
writedate2
	mov BYTE al,[ds:si]
	mov BYTE [es:di],al
	inc si
	inc di
	loop writedate2
	mov BYTE [es:di],'#'
	inc di
	mov BYTE [es:di],'0'
	inc di
	mov BYTE [es:di],'#'
	inc di
	mov BYTE [es:di],'0'
	inc di
	mov BYTE [es:di],'#'
	inc di
	mov BYTE [es:di],'0'
	inc di
	mov BYTE [es:di],'#'
	inc di
	mov BYTE [es:di],'0'
	inc di
	mov BYTE [es:di],'#'
	inc di
	mov BYTE [es:di],'0'
	inc di
	mov BYTE [es:di],'#'
	inc di
	mov BYTE [es:di],'0'
	inc di
	mov BYTE [es:di],'#'
	inc di
	mov BYTE [es:di],'1'
	inc di
	mov BYTE [es:di],'2'
	inc di
	mov BYTE [es:di],'3'
	inc di
	mov BYTE [es:di],'4'
	inc di
```

```
      mov BYTE [es:di],'5'
      inc di
      mov BYTE [es:di],'#'
      inc di
      pop si
      pop ds
      mov si,34
      mov BYTE al,[ds:si]
      mov BYTE [es:di],al
      inc di
      mov cx,di
      mov di,0
userdisp
      mov BYTE al,[es:di]
      inc di
      dispchar
      loop userdisp
;-------------------------------------------------------------
searchagain
      mov ax,500h         ;READING THE NEXT FILE ENTRY SECTOR
      mov es,ax
      mov di,0
      xor bx,bx
      mov ah,2
      mov al,1
      mov ch,0
      mov cl,1
      mov dh,1
      mov dl,0
      int 13h
      mov di,010ah
      mov BYTE ch,[es:di]
      mov di,010ch
      mov BYTE dh,[es:di]
      mov di,010eh
      mov BYTE cl,[es:di]
      mov ax,3000h
      mov es,ax
      mov di,0
      xor bx,bx
      mov al,1
      mov ah,02
      mov dl,0
      int 13h
      mov di,0
      mov cx,512
```

```
next3   mov BYTE al,[es:di]
       cmp al,255
       je writeentry
       inc di
       loopnz next3
       jmp nospace
writeentry
       cmp di,408
       jne proceed
       push di
       mov di,510
       mov BYTE [es:di],'#'
       inc di
       mov BYTE [es:di],'#'
       pop di
proceed mov ax,2000h
       mov ds,ax
       mov cx,102
       mov si,0
writerecord
       mov BYTE al,[ds:si]
       mov BYTE [es:di],al
       inc di
       inc si
       loopnz writerecord
       cmp di,510
       je endofsect
       mov BYTE [es:di],255
endofsect nop

       mov ax,500h          ;READING THE NEXT FILE ENTRY SECTOR
       mov es,ax
       mov di,0
       mov di,010ah
       mov BYTE ch,[es:di]
       mov di,010ch
       mov BYTE dh,[es:di]
       mov di,010eh
       mov BYTE cl,[es:di]
       mov ax,3000h          ;WRITING THE UPDATED BUFFER TO FILE RECORD
       mov es,ax            ;ENTRIES SECTOR
       mov di,0
       mov di,0
       xor bx,bx
       mov al,1
       mov ah,03
```

```
      mov dl,0
      int 13h

      jmp exit1
nospace mov ax,500h
      mov es,ax
      mov di,0
      mov di,010ah
      mov BYTE ah,[es:di]
      mov di,010ch
      mov BYTE ch,[es:di]
      mov di,010eh
      mov BYTE dh,[es:di]
      mov di,0122h
      mov BYTE al,[es:di]
      mov di,0124h
      mov BYTE cl,[es:di]
      mov di,0126h
      mov BYTE dl,[es:di]
      cmp dh,dl
      jne skiphead
      cmp ch,cl
      jne skiptrack
      sub al,ah
      cmp al,1
      je diskfull
skiphead  nop
skiptrack nop
      mov di,010ah
      mov BYTE ah,[es:di]
      mov di,010ch
      mov BYTE ch,[es:di]
      mov di,010eh
      mov BYTE dh,[es:di]
      cmp dh,18
      jl nextsect
      cmp ch,1
      jl nexthead
      mov dh,1
      mov ch,0
      inc ah
      jmp updatefile
nextsect
      inc dh
      jmp updatefile
nexthead
```

```
      mov ch,1

      mov dh,1
updatefile
      mov di,010ah
      mov BYTE [es:di],ah
      mov di,010ch
      mov BYTE [es:di],ch
      mov di,010eh
      mov BYTE [es:di],dh
;--------------------------UPDATING FILE ENTRY SECTOR

      mov di,0
      xor bx,bx
      mov ah,3
      mov al,1
      mov ch,0
      mov cl,1
      mov dh,1
      mov dl,0
      int 13h
      jmp searchagain
;---------------------------------------------------
diskfull
      pop si
      pop di
      pop es
      pop ds

      writestring fullalert,36,22,10
      keypress
      jmp bye
;---------------------------------------------------
exit1   pop si
      pop di
      pop es
      pop ds
      writestring success,39,22,10
      jmp bye
exit3
      pop si
      pop di
      pop es
      pop ds

      pop si
      pop di
      pop es
      pop ds
      writestring changefilename,40,22,10
bye
      keypress
      ;mov ax,$4c
      ;int 21h
      retf

[SECTION .data]
      message db 'THIS COMMAND CREATES ONLY FILES....'
      filename db   'FILENAME (33 CHARACTERS ONLY) :'
      permission db 'PERMISSIONS PRESS 1 FOR TRUE 0 FOR FALSE :'
      ronly  db 'READ ONLY'
      wonly  db 'WRITE ONLY'
      xonly  db 'EXECUTABLE ONLY'
      access db 'INACCESSIBLE'
      system db 'SYSTEM FILE'
      hidden db 'HIDDEN FILE'
      ascii  db 'ASCII FILE'
      username db 'anon          '
      fullalert db 'SORRY DISK IS FULL...              '
      success db 'FILE SUCCESSFULLY CREATED...            '
      changefilename db 'FILE ALREADY PRESENT CHANGE THE FILENAME'
[section .bss]
      maxtrack resb 1
      maxhead  resb 1
      maxsect  resb 1
      curtrack resb 1
      curhead  resb 1
      cursect  resb 1



;WORKING PROGRAM WHICH DELETES USER SPECIFIED FILE ...
[org 0x0100]
[section .text]
   %macro writestring 4       ;macro to write the string
      mov ax,%1             ;the offset value to bp thru ax
      mov bp,ax
      mov ah,13h             ;move function number to ah
      mov al,1             ;move write mode to al
      mov bh,0              ;move vdu page no to  bh
      mov bl,14             ;move attribute to bl
      mov cx,%2               ;move the stringlength to cx as parameter2
      mov dh,%3               ;move row to dh reg
      mov dl,%4            ;move col to dl
      int 10h             ;call the interrupt
   %endmacro
   push es
   push ds
push si
push di

push es
push ds
push si
push di
       mov ax,600h
       mov es,ax
       mov di,0
       mov cx,13
       mov si,username
name    mov BYTE al,[ds:si]
       mov BYTE [es:di],al
       inc si
       inc di
       loopnz name

       mov ax,10h
       mov es,ax
       mov di,0
getkey  xor ah,ah
       int 16h
       cmp al,13
       je endofstr
       mov [es:di],al
       inc di
       mov ah,14
       mov bh,0
       int 10h
       jmp getkey

endofstr  mov ax,33
        mov cx,di
        sub ax,cx
        mov cx,ax
fillspace mov al,20h
        mov [es:di],al
        inc di
        loop fillspace
;-----------------------------------------
```

```
      mov ax,12h
      mov es,ax
      mov di,0
      xor bx,bx
      mov ah,2
      mov al,1
      mov ch,0
      mov cl,1
      mov dh,1
      mov dl,0
      int 13h
      mov di,0
      mov di,010ah
      mov BYTE al,[es:di]

      mov BYTE [maxtrack],al
      mov di,010ch
      mov BYTE al,[es:di]

      mov BYTE [maxhead],al
      mov di,010eh
      mov BYTE al,[es:di]

      mov BYTE [maxsect],al
      mov BYTE [curtrack],0
      mov BYTE [curhead],1
      mov BYTE [cursect],2
      pop di
      pop si
      pop ds
      pop es
;-----------------------------------------------------------------
nextsector push es
        push ds
        push si
        push di
        mov ax,100h
        mov es,ax
        mov bx,0
        mov ah,2
        mov al,1
        mov ch,[curtrack]
        mov cl,[cursect]
        mov dh,[curhead]
        mov dl,0
        int 13h
        mov ax,300h
        mov fs,ax
        mov si,0
        mov BYTE al,[curtrack]
        mov BYTE [fs:si],al
        inc si
        mov BYTE al,[curhead]
        mov BYTE [fs:si],al
        inc si
        mov BYTE al,[cursect]
        mov BYTE [fs:si],al

;----------------------
       mov di,0
;----------------------------------------
mov ax,100h
mov ds,ax
mov ax,10h
mov es,ax
mov si,0
mov di,0
cmpnext
inc si
inc si
            ;comparing file name
            mov di,0
            mov cx,33
            repe cmpsb
            cmp cx,0
            jne nequal
            ;comparing user name
            inc si
            mov ax,600h
            mov es,ax
            mov di,0
            mov cx,13
            repe cmpsb
            cmp cx,0
            je key

nequal
     cmp si,445
     jge npres
     mov ax,si
     mov bl,102
     div bl
     inc al
     mov bl,102
     mul bl
     mov si,ax
     jmp cmpnext


key
   mov ax,100h
   mov ds,ax
   mov ax,si
   mov bl,102
   div bl
   mul bl
;------------------------roll back to start of record and display it
   mov si,ax
   mov cx,102
   push es
   push di
   mov ax,20h
   mov es,ax
   mov di,0

disprec    mov al,[ds:si]
        mov ah,14
        mov bh,0
        int 10h
        mov [es:di],al
        inc di
        inc si
        loop disprec
   pop di
   pop es
   jmp near exit
;------------------------------------------display record
npres
;----------------------------------------
       pop di
       pop si
       pop ds
       pop es

       mov BYTE al,[cursect]
       cmp al,18
       jl near incsect
       mov BYTE [cursect],1
```

```
        mov BYTE al,[curhead]
        cmp al,1
        jl near inchead
        mov BYTE [curhead],0
        mov BYTE al,[curtrack]
        inc al
        mov BYTE [curtrack],al
        jmp near next
inchead  mov BYTE [curhead],1
        jmp near next1
incsect  mov BYTE al,[cursect]
        inc al
        mov BYTE [cursect],al
next     nop
next1    mov BYTE al,[curtrack]
        mov BYTE ah,[maxtrack]
        cmp al,ah
        jl near nextsector
        mov BYTE al,[curhead]
        mov BYTE ah,[maxhead]
        cmp al,ah
        jl near nextsector
        mov BYTE al,[cursect]
        mov BYTE ah,[maxsect]
        cmp al,ah
        jle near nextsector
;----------------------------------------------------------------
exit
mov ax,20h
mov es,ax
mov di,0
cmp BYTE [es:di],'#'
jne near exit2

;DELETION  OF RECORD ENTRY BEGINS......................
delete     mov ax,100h
    mov ds,ax
    mov ax,10h
    mov es,ax
    mov si,0
    mov di,0
cmpnext1        inc si
            inc si
            mov di,0
            mov cx,33
            repe cmpsb
```

```
          cmp cx,0
          jne nequal1
          ;comparing user name
          inc si
          mov ax,600h
          mov es,ax
          mov di,0
          mov cx,13
          repe cmpsb
          cmp cx,0
          je key1
nequal1
      mov ax,si
      mov bl,102
      div bl
      inc al
      mov bl,102
      mul bl
      mov si,ax
      jmp cmpnext1

 key1 mov ax,si
     mov bl,102
     div bl
     mul bl
     mov si,ax
     mov cx,102
 delrec  mov BYTE [ds:si],' '
        inc si
        loopnz delrec
      mov ax,300h
      mov fs,ax
      mov si,0

      mov ax,100h
      mov es,ax
      mov di,0
      mov bx,0
      mov ah,3
      mov al,1
      mov ch,[fs:si]
      inc si
      mov dh,[fs:si]
      inc si
      mov cl,[fs:si]
      mov dl,0
```

```
   int 13h
;----------END OF DELETION-------------

      pop di
      pop si
      pop ds
      pop es
      writestring message2,12,20,10
      jmp bye
exit2   pop di
      pop si
      pop ds
      pop es
      writestring message1,16,20,10
bye     xor ah,ah
      int 16h
      ;mov ax,$4c
      ;int 21h
      retf
;-------------------------------------------------------------------
[section .data]
username db 'anon        '
message1 db 'FILE NOT PRESENT'
message2 db 'FILE DELETED'
[section .bss]
maxtrack resb 1
maxhead  resb 1
maxsect  resb 1
curtrack resb 1
curhead  resb 1
cursect  resb 1


;PROGRAM TO LIST FILES OF THE CURRENT USER
[org 0x0100]
[section .text]
jmp near start
%macro scrollwindow 0
   mov ah,06h
   mov al,0
   mov bh,0
   mov ch,0
   mov cl,0
   mov dh,25
   mov dl,79
   int 10h
```

```
%endmacro
%macro writestring 4       ;macro to write the string
  mov ax,%1              ;the offset value to bp thru ax
  mov bp,ax
  mov ah,13h             ;move function number to ah
  mov al,1              ;move write mode to al
  mov bh,0               ;move vdu page no to  bh
  mov bl,14              ;move attribute to bl
  mov cx,%2                ;move the stringlength to cx as parameter2
  mov dh,%3                ;move row to dh reg
  mov dl,%4               ;move col to dl
  int 10h             ;call the interrupt
%endmacro
%macro incline 0
       call inccursor
       mov ah,3
       xor bh,bh
       int 10h
       mov ah,02
       xor bh,bh
       xor dl,dl
       int 10h
%endmacro
%macro keypress 0
xor ah,ah
int 16h
%endmacro
inccursor:      ; MACRO TO INCREMENT ROW IN THE CURRENT COLUMN
      mov ah,3
      mov bh,0
      int 10h
      xor dl,dl
      cmp dh,24
      add dh,1
      jl near pro
      keypress
      scrollwindow
pro   mov ah,2
      mov bh,0
      int 10h
      ret
start:
push es
push ds
push si
push di
```

```
        writestring title,25,1,25
        incline
        mov ax,60h
        mov es,ax
        mov di,0
        mov cx,13
        mov si,username
name    mov BYTE al,[ds:si]
        mov BYTE [es:di],al
        inc si
        inc di
        loopnz name

        mov ax,12h
        mov es,ax
        mov di,0
        xor bx,bx
        mov ah,2
        mov al,1
        mov ch,0
        mov cl,1
        mov dh,1
        mov dl,0
        int 13h
        mov di,010ah
        mov BYTE al,[es:di]
        mov BYTE [maxtrack],al
        mov di,010ch
        mov BYTE al,[es:di]
        mov BYTE [maxhead],al
        mov di,010eh
        mov BYTE al,[es:di]
        mov BYTE [maxsect],al
        mov BYTE [curtrack],0
        mov BYTE [curhead],1
        mov BYTE [cursect],2
        pop di
        pop si
        pop ds
        pop es

 nextsector push es
           push ds
           push si
           push di
           mov ax,100h
        mov es,ax
        mov bx,0
        mov ah,2
        mov al,1
        mov ch,[curtrack]
        mov cl,[cursect]
        mov dh,[curhead]
        mov dl,0
        int 13h
;-----------------------------------------
       mov ax,100h
       mov ds,ax
       mov ax,60h
       mov es,ax
       mov si,0
       mov di,0
cmpnext
            mov ax,si
            add ax,36
            mov si,ax
            mov di,0
            mov cx,13
            repe cmpsb
            cmp cx,0
            jne nequal
            jmp key


nequal
        cmp si,445
        jge near npres
        mov ax,si
        mov bl,102
        div bl
        inc al
        mov bl,102
        mul bl
        mov si,ax
        jmp cmpnext

key
        mov ax,100h
        mov ds,ax
        mov ax,si
        mov bl,102
        div bl
        mul bl
;------------------------roll back to start of record and display it
   mov si,ax
   mov cx,102
   push es
   push di
   mov ax,20h
   mov es,ax
   mov di,2

disprec    mov al,[ds:si]
         mov ah,14
         mov bh,0
         int 10h
         mov [es:di],al
         inc di
         inc si
         loop disprec
         pop di
         pop es
         ;incline
         incline
         jmp near cmpnext
;-----------------------------------------display record
npres
;----------------------------------------
        pop di
        pop si
        pop ds
        pop es

        mov BYTE al,[cursect]
        cmp al,18
        jl near incsect
        mov BYTE [cursect],1
        mov BYTE al,[curhead]
        cmp al,1
        jl near inchead
        mov BYTE [curhead],0
        mov BYTE al,[curtrack]
        inc al
        mov BYTE [curtrack],al
        jmp near next
inchead  mov BYTE [curhead],1
        jmp near next1
incsect  mov BYTE al,[cursect]
```

```
       inc al
       mov BYTE [cursect],al
next    nop
next1    mov BYTE al,[curtrack]
       mov BYTE ah,[maxtrack]
       cmp al,ah
       jl near nextsector
       mov BYTE al,[curhead]
       mov BYTE ah,[maxhead]
       cmp al,ah
       jl near nextsector
       mov BYTE al,[cursect]
       mov BYTE ah,[maxsect]
       cmp al,ah
       jle near nextsector
       writestring message,25,24,25
       keypress
       mov ax,$4c
       int 21h
[section .data]
username db 'anon          '
title   db 'FILE LIST OF CURRENT USER'
message  db 'PRESS ANY KEY TO CONTINUE'
[section .bss]
maxtrack resb 1
maxhead  resb 1
maxsect  resb 1
curtrack resb 1
curhead  resb 1
cursect  resb 1


;WORKING PROGRAM TO ENCRYPT A FILE.......

[org 0x0100]
[section .text]
%macro writestring 4      ;macro to write the string
   mov ax,%1             ;the offset value to bp thru ax
   mov bp,ax
   mov ah,13h             ;move function number to ah
   mov al,1             ;move write mode to al
   mov bh,0             ;move vdu page no to  bh
   mov bl,14            ;move attribute to bl
   mov cx,%2             ;move the stringlength to cx as parameter2
   mov dh,%3             ;move row to dh reg
   mov dl,%4             ;move col to dl
```

```
  int 10h              ;call the interrupt
%endmacro

push es
push ds
push si
push di
push es
push ds
push si
push di
        mov ax,60h
        mov es,ax
        mov di,0
        mov cx,13
        mov si,username
name    mov BYTE al,[ds:si]
        mov BYTE [es:di],al
        inc si
        inc di
        loopnz name
        mov ax,70h         ;INITIALISING BUFFER WITH USER ENCRYPTION KEY
        mov fs,ax
        mov di,0
        mov BYTE [fs:di],3

        mov ax,10h
        mov es,ax
        mov di,0
getkey  xor ah,ah
        int 16h
        cmp al,13
        je endofstr
        mov [es:di],al
        inc di
        mov ah,14
        mov bh,0
        int 10h
        jmp getkey

endofstr  mov ax,33
          mov cx,di
          sub ax,cx
          mov cx,ax
fillspace mov al,20h
          mov [es:di],al
        inc di
        loop fillspace
;---------------------------------------
       mov ax,12h
       mov es,ax
       mov di,0
       xor bx,bx
       mov ah,2
       mov al,1
       mov ch,0
       mov cl,1
       mov dh,1
       mov dl,0
       int 13h
       mov di,0
       mov di,010ah
       mov BYTE al,[es:di]
       mov BYTE [maxtrack],al
       mov di,010ch
       mov BYTE al,[es:di]
       mov BYTE [maxhead],al
       mov di,010eh
       mov BYTE al,[es:di]
       mov BYTE [maxsect],al
       mov BYTE [curtrack],0
       mov BYTE [curhead],1
       mov BYTE [cursect],2
       pop di
       pop si
       pop ds
       pop es
;---------------------------------------------------------------
nextsector push es
        push ds
        push si
        push di
        mov ax,100h
        mov es,ax
        mov bx,0
        mov ah,2
        mov al,1
        mov ch,[curtrack]
        mov cl,[cursect]
        mov dh,[curhead]
        mov dl,0
        int 13h
```

```
;-----------------------------------------------
mov ax,100h
mov ds,ax
mov ax,10h
mov es,ax
mov si,0
mov di,0
cmpnext
inc si
inc si
            mov di,0
            mov cx,33
            repe cmpsb
            cmp cx,0
            jne nequal
            ;comparing user name
            inc si
            mov ax,60h
            mov es,ax
            mov di,0
            mov cx,13
            repe cmpsb
            cmp cx,0
            je key

nequal
     cmp si,445
     jge npres
     mov ax,si
     mov bl,102
     div bl
     inc al
     mov bl,102
     mul bl
     mov si,ax
     jmp cmpnext


key
   mov ax,100h
   mov ds,ax
   mov ax,si
   mov bl,102
   div bl
   mul bl
;----------------roll back to start of record and display it
```

```
   mov si,ax
   mov cx,102
   push es
   push di
   mov ax,20h
   mov es,ax
   mov di,0

disprec    mov al,[ds:si]
   mov ah,14
   mov bh,0
   int 10h
   mov [es:di],al
   inc di
   inc si
   loop disprec
   pop di
   pop es
   jmp near exit
;-----------------------------DISPLAY RECORD----------
npres
        pop di
        pop si
        pop ds
        pop es

        mov BYTE al,[cursect]
        cmp al,18
        jl near incsect
        mov BYTE [cursect],1
        mov BYTE al,[curhead]
        cmp al,1
        jl near inchead
        mov BYTE [curhead],0
        mov BYTE al,[curtrack]
        inc al
        mov BYTE [curtrack],al
        jmp near next
 inchead  mov BYTE [curhead],1
        jmp near next1
 incsect  mov BYTE al,[cursect]
        inc al
        mov BYTE [cursect],al
 next    nop
 next1    mov BYTE al,[curtrack]
        mov BYTE ah,[maxtrack]
        cmp al,ah
        jl near nextsector
        mov BYTE al,[curhead]
        mov BYTE ah,[maxhead]
        cmp al,ah
        jl near nextsector
        mov BYTE al,[cursect]
        mov BYTE ah,[maxsect]
        cmp al,ah
        jle near nextsector
;-----------------------------------------------------------------
exit
mov ax,20h
mov es,ax
mov di,0
cmp BYTE [es:di],'#'
jne near exit2

mov di,83                ;INITIALISE FOR READ FILE
mov BYTE ch,[es:di]
inc di
inc di
mov BYTE dh,[es:di]
inc di
inc di
mov BYTE cl,[es:di]
mov di,93
mov BYTE ah,[es:di]
mov di,87
mov BYTE al,[es:di]
cmp ah,al
;----------------------------------------------------
;------------------FILE LOCATED IN SAME HEADS  TRACKS
jl sect
sub ah,al
inc ah
mov al,ah
mov ah,2
mov bx,ax
mov ax,2000h
mov es,ax
mov ax,bx
xor bx,bx
int 13h
jmp dic
;---------------------FILE LOCATED IN DIFFERENT TRACKS OR HEADS
```

```
sect mov ah,18
    sub ah,al
    inc ah
    mov al,ah
    mov ah,2
    mov bx,ax
    mov ax,2000h
    mov es,ax
    mov ax,bx
    xor bx,bx
    int 13h
;---------------READ FIRST PART FULLY & PERFORM ENCRYPTION
   mov si,0
   mov di,0
   mov cx,1024
   mov BYTE dh,[fs:si]
disp2 mov BYTE al,[es:di]
     add al,dh
     mov BYTE [es:di],al
     inc di
     loop disp2
    mov ax,20h
    mov es,ax
    mov di,0
    mov di,83            ;INITIALISE FOR WRITING ENCRYPTED DATA TO FILE
    mov BYTE ch,[es:di]
    inc di
    inc di
    mov BYTE dh,[es:di]
    inc di
    inc di
    mov BYTE cl,[es:di]
    mov di,93
    mov BYTE ah,[es:di]
    mov di,87
    mov BYTE al,[es:di]
    mov ah,18
    sub ah,al
    inc ah
    mov al,ah
    mov ah,3
    mov bx,ax
    mov ax,2000h
    mov di,0
    mov es,ax
```

```
  mov ax,bx
  xor bx,bx
  int 13h
;--------------------------------------------------------
  mov ax,20h   ;INITIALISE FOR THE SECOND PART OF FILE
  mov es,ax
  mov di,89
  mov BYTE ch,[es:di]
  inc di
  inc di
  mov BYTE dh,[es:di]
  mov cl,1
  inc di
  inc di
  mov BYTE al,[es:di]
  mov ah,2
  mov bx,ax
  mov ax,2001h
  mov es,ax
  mov ax,bx
  xor bx,bx
  int 13h
;---------- READ THE CONTENTS & PERFORM ENCRYPTION
  mov si,0
  mov BYTE dh,[fs:si]
  mov di,0

disp1 mov BYTE al,[es:di]
   cmp al,255
   je outloop
   add al,dh
   mov BYTE [es:di],al
   inc di
   jmp disp1
outloop
    mov ax,20h   ;INITIALISE FOR THE SECOND PART OF FILE
    mov es,ax
    mov di,89
    mov BYTE ch,[es:di]
    inc di
    inc di
    mov BYTE dh,[es:di]
    mov cl,1
    inc di
    inc di
    mov BYTE al,[es:di]
    mov ah,3
    mov bx,ax
    mov ax,2001h
    mov es,ax
    mov ax,bx
    xor bx,bx
    int 13h
    jmp exit1

;----------- READ THE CONTENTS & PERFORM ENCRYPTION
dic  mov di,0
    mov si,0
    mov BYTE dh,[fs:di]
disp mov BYTE al,[es:di]
    cmp al,255
    je loopout
    add al,dh
    mov BYTE [es:di],al
    inc di
    jmp disp
loopout  mov ax,20h
       mov es,ax
       mov di,0
       mov di,83            ;INITIALISE TO WRITE ENCRYPTED DATA TO FILE
       mov BYTE ch,[es:di]
       inc di
       inc di
       mov BYTE dh,[es:di]
       inc di
       inc di
       mov BYTE cl,[es:di]
       mov di,93
       mov BYTE ah,[es:di]
       mov di,87
       mov BYTE al,[es:di]
       sub ah,al
       inc ah
       mov al,ah
       mov ah,3
       mov bx,ax
       mov ax,2000h
       mov di,0
       mov es,ax
       mov ax,bx
       xor bx,bx
       int 13h
```

```
      jmp exit1

exit2   pop di
      pop si
      pop ds
      pop es
      writestring nofile,17,20,10
      jmp bye
exit1   pop di
      pop si
      pop ds
      pop es
      writestring success,35,20,10
bye     xor ah,ah
      int 16h
      ;mov ax,$4c
      ;int 21h
      retf

;----------------------------------------------------------------------
[section .data]
username db 'anon         '
nofile db 'FILE NOT FOUND...'
success db 'ENCRYPTION SUCCESSFULLY COMPLETED...'
[section .bss]
maxtrack resb 1
maxhead  resb 1
maxsect  resb 1
curtrack resb 1
curhead  resb 1
cursect  resb 1



  ;WORKING PROGRAM TO DECRYPT A FILE.......
  [org 0x0100]
  [section .text]
  %macro writestring 4       ;macro to write the string
    mov ax,%1              ;the offset value to bp thru ax
    mov bp,ax
    mov ah,13h              ;move function number to ah
    mov al,1               ;move write mode to al
    mov bh,0                ;move vdu page no to  bh
    mov bl,14               ;move attribute to bl
    mov cx,%2                ;move the stringlength to cx as parameter2
  mov dh,%3              ;move row to dh reg
  mov dl,%4             ;move col to dl
```

```
   int 10h             ;call the interrupt
%endmacro

push es
push ds
push si
push di
push es
push ds
push si
push di
       mov ax,60h
       mov es,ax
       mov di,0
       mov cx,13
       mov si,username
name    mov BYTE al,[ds:si]
       mov BYTE [es:di],al
       inc si
       inc di
       loopnz name
       mov ax,70h          ;INITIALISING BUFFER WITH USER DECRYPTION KEY
       mov fs,ax
       mov di,0
       mov BYTE [fs:di],3

       mov ax,10h
       mov es,ax
       mov di,0
getkey  xor ah,ah
       int 16h
       cmp al,13
       je endofstr
       mov [es:di],al
       inc di
       mov ah,14
       mov bh,0
       int 10h
       jmp getkey

endofstr  mov ax,33
        mov cx,di
        sub ax,cx
        mov cx,ax
fillspace mov al,20h
        mov [es:di],al
        inc di
        loop fillspace
;-----------------------------------------
       mov ax,12h
       mov es,ax
       mov di,0
       xor bx,bx
       mov ah,2
       mov al,1
       mov ch,0
       mov cl,1
       mov dh,1
       mov dl,0
       int 13h
       mov di,0
       mov di,010ah
       mov BYTE al,[es:di]
       mov BYTE [maxtrack],al
       mov di,010ch
       mov BYTE al,[es:di]
       mov BYTE [maxhead],al
       mov di,010eh
       mov BYTE al,[es:di]
       mov BYTE [maxsect],al
       mov BYTE [curtrack],0
       mov BYTE [curhead],1
       mov BYTE [cursect],2
       pop di
       pop si
       pop ds
       pop es
;----------------------------------------------------------------
nextsector push es
        push ds
        push si
        push di
        mov ax,100h
        mov es,ax
        mov bx,0
        mov ah,2
        mov al,1
        mov ch,[curtrack]
        mov cl,[cursect]
        mov dh,[curhead]
        mov dl,0
        int 13h
```

```
;---------------------------------------------------
mov ax,100h
mov ds,ax
mov ax,10h
mov es,ax
mov si,0
mov di,0
cmpnext
inc si
inc si
                mov di,0
                mov cx,33
                repe cmpsb
                cmp cx,0
                jne nequal
                ;comparing user name
                inc si
                mov ax,60h
                mov es,ax
                mov di,0
                mov cx,13
                repe cmpsb
                cmp cx,0
                je key

nequal
      cmp si,445
      jge npres
      mov ax,si
      mov bl,102
      div bl
      inc al
      mov bl,102
      mul bl
      mov si,ax
      jmp cmpnext


key
   mov ax,100h
   mov ds,ax
   mov ax,si
   mov bl,102
   div bl
   mul bl
;----------------roll back to start of record and display it
```

```
   mov si,ax
   mov cx,102
   push es
   push di
   mov ax,20h
   mov es,ax
   mov di,0

disprec    mov al,[ds:si]
   mov ah,14
   mov bh,0
   int 10h
   mov [es:di],al
   inc di
   inc si
   loop disprec
   pop di
   pop es
   jmp near exit
;----------------------------DISPLAY RECORD----------
npres
       pop di
       pop si
       pop ds
       pop es

       mov BYTE al,[cursect]
       cmp al,18
       jl near incsect
       mov BYTE [cursect],1
       mov BYTE al,[curhead]
       cmp al,1
       jl near inchead
       mov BYTE [curhead],0
       mov BYTE al,[curtrack]
       inc al
       mov BYTE [curtrack],al
       jmp near next
inchead  mov BYTE [curhead],1
       jmp near next1
incsect  mov BYTE al,[cursect]
       inc al
       mov BYTE [cursect],al
next    nop
next1    mov BYTE al,[curtrack]
       mov BYTE ah,[maxtrack]
```

```
        cmp al,ah
        jl near nextsector
        mov BYTE al,[curhead]
        mov BYTE ah,[maxhead]
        cmp al,ah
        jl near nextsector
        mov BYTE al,[cursect]
        mov BYTE ah,[maxsect]
        cmp al,ah
        jle near nextsector
;---------------------------------------------------------------
exit
mov ax,20h
mov es,ax
mov di,0
cmp BYTE [es:di],'#'
jne near exit2

mov di,83                  ;INITIALISE FOR READ FILE
mov BYTE ch,[es:di]
inc di
inc di
mov BYTE dh,[es:di]
inc di
inc di
mov BYTE cl,[es:di]
mov di,93
mov BYTE ah,[es:di]
mov di,87
mov BYTE al,[es:di]
cmp ah,al
;---------------------------------------------------
;-------------------FILE LOCATED IN SAME HEADS  TRACKS
jl sect
sub ah,al
inc ah
mov al,ah
mov ah,2
mov bx,ax
mov ax,2000h
mov es,ax
mov ax,bx
xor bx,bx
int 13h
jmp dic
;---------------------FILE LOCATED IN DIFFERENT TRACKS OR HEADS
```

```
sect mov ah,18
    sub ah,al
    inc ah
    mov al,ah
    mov ah,2
    mov bx,ax
    mov ax,2000h
    mov es,ax
    mov ax,bx
    xor bx,bx
    int 13h
;---------------READ FIRST PART FULLY & PERFORM DECRYPTION
   mov si,0
   mov di,0
   mov cx,1024
   mov BYTE dh,[fs:si]
disp2 mov BYTE al,[es:di]
     sub al,dh
     mov BYTE [es:di],al
     inc di
     loop disp2
   mov ax,20h
   mov es,ax
   mov di,0
   mov di,83          ;INITIALISE FOR WRITING ENCRYPTED DATA TO FILE
   mov BYTE ch,[es:di]
   inc di
   inc di
   mov BYTE dh,[es:di]
   inc di
   inc di
   mov BYTE cl,[es:di]
   mov di,93
   mov BYTE ah,[es:di]
   mov di,87
   mov BYTE al,[es:di]
   mov ah,18
   sub ah,al
   inc ah
   mov al,ah
   mov ah,3
   mov bx,ax
   mov ax,2000h
   mov di,0
   mov es,ax
   mov ax,bx
   xor bx,bx
   int 13h
;-----------------------------------------------------
   mov ax,20h   ;INITIALISE FOR THE SECOND PART OF FILE
   mov es,ax
   mov di,89
   mov BYTE ch,[es:di]
   inc di
   inc di
   mov BYTE dh,[es:di]
   mov cl,1
   inc di
   inc di
   mov BYTE al,[es:di]
   mov ah,2
   mov bx,ax
   mov ax,2001h
   mov es,ax
   mov ax,bx
   xor bx,bx
   int 13h
;---------- READ THE CONTENTS & PERFORM DECRYPTION
   mov si,0
   mov BYTE dh,[fs:si]
   mov di,0

disp1 mov BYTE al,[es:di]
    cmp al,255
    je outloop
    sub al,dh
    mov BYTE [es:di],al
    inc di
    jmp disp1
outloop
     mov ax,20h   ;INITIALISE FOR THE SECOND PART OF FILE
     mov es,ax
     mov di,89
     mov BYTE ch,[es:di]
     inc di
     inc di
     mov BYTE dh,[es:di]
     mov cl,1
     inc di
     inc di
     mov BYTE al,[es:di]
    mov ah,3
    mov bx,ax
    mov ax,2001h
    mov es,ax
    mov ax,bx
    xor bx,bx
    int 13h
    jmp exit1

;----------- READ THE CONTENTS & PERFORM DECRYPTION
dic  mov di,0
    mov si,0
    mov BYTE dh,[fs:di]
disp mov BYTE al,[es:di]
    cmp al,255
    je loopout
    sub al,dh
    mov BYTE [es:di],al
    inc di
    jmp disp
loopout  mov ax,20h
        mov es,ax
        mov di,0
        mov di,83              ;INITIALISE TO WRITE ENCRYPTED DATA TO FILE
        mov BYTE ch,[es:di]
        inc di
        inc di
        mov BYTE dh,[es:di]
        inc di
        inc di
        mov BYTE cl,[es:di]
        mov di,93
        mov BYTE ah,[es:di]
        mov di,87
        mov BYTE al,[es:di]
        sub ah,al
        inc ah
        mov al,ah
        mov ah,3
        mov bx,ax
        mov ax,2000h
        mov di,0
        mov es,ax
        mov ax,bx
        xor bx,bx
        int 13h
```

```
      jmp exit1

exit2   pop di
      pop si
      pop ds
      pop es
      writestring nofile,17,20,10
      jmp bye
exit1   pop di
      pop si
      pop ds
      pop es
      writestring success,35,20,10
bye     xor ah,ah
      int 16h
      ;mov ax,$4c
      ;int 21h
      retf

;-----------------------------------------------------------------------
[section .data]
username db 'anon         '
nofile db 'FILE NOT FOUND...'
success db 'DECRYPTION SUCCESSFULLY COMPLETED...'
[section .bss]
maxtrack resb 1
maxhead  resb 1
maxsect  resb 1
curtrack resb 1
curhead  resb 1
cursect  resb 1




;PROGRAM TO PRINT USER SPECIFIED FILE
[org 0x0100]
[section .text]
%macro writestring 4      ;macro to write the string
   mov ax,%1            ;the offset value to bp thru ax
   mov bp,ax
   mov ah,13h           ;move function number to ah
   mov al,1            ;move write mode to al
   mov bh,0             ;move vdu page no to  bh
   mov bl,14            ;move attribute to bl
   mov cx,%2            ;move the stringlength to cx as parameter2
   mov dh,%3            ;move row to dh reg
   mov dl,%4            ;move col to dl
   int 10h            ;call the interrupt
%endmacro
%macro dispchar 0
mov ah,14
mov bh,0
int 10h
%endmacro
push es
push ds
push si
push di
push es
push ds
push si
push di

        mov ax,60h
        mov es,ax
        mov di,0
        mov cx,13
        mov si,username
 name     mov BYTE al,[ds:si]
        mov BYTE [es:di],al
        inc si
        inc di
        loopnz name

        mov ax,10h
        mov es,ax
        mov di,0
 getkey   xor ah,ah
        int 16h
        cmp al,13
        je endofstr
        mov [es:di],al
        inc di
        mov ah,14
        mov bh,0
        int 10h
        jmp getkey

 endofstr  mov ax,33
         mov cx,di
         sub ax,cx
         mov cx,ax
```

```
fillspace mov al,20h
        mov [es:di],al
        inc di
        loop fillspace
;----------------------------------------
       mov ax,12h
       mov es,ax
       mov di,0
       xor bx,bx
       mov ah,2
       mov al,1
       mov ch,0
       mov cl,1
       mov dh,1
       mov dl,0
       int 13h
       mov di,0
       mov di,010ah
       mov BYTE al,[es:di]
       mov BYTE [maxtrack],al
       mov di,010ch
       mov BYTE al,[es:di]
       mov BYTE [maxhead],al
       mov di,010eh
       mov BYTE al,[es:di]
       mov BYTE [maxsect],al
       mov BYTE [curtrack],0
       mov BYTE [curhead],1
       mov BYTE [cursect],2
       pop di
       pop si
       pop ds
       pop es
;-----------------------------------------------------------------
nextsector push es
          push ds
          push si
          push di
          mov ax,100h
          mov es,ax
          mov bx,0
          mov ah,2
          mov al,1
          mov ch,[curtrack]
          mov cl,[cursect]
          mov dh,[curhead]
        mov dl,0
        int 13h
;-----------------------
       mov di,0
;-------------------------------------------
mov ax,100h
mov ds,ax
mov ax,10h
mov es,ax
mov si,0
mov di,0
cmpnext
inc si
inc si
            mov di,0
            mov cx,33
            repe cmpsb
            cmp cx,0
            jne nequal
            ;comparing user name
            inc si
            mov ax,60h
            mov es,ax
            mov di,0
            mov cx,13
            repe cmpsb
            cmp cx,0
            je key
nequal
      cmp si,445
      jge npres
      mov ax,si
      mov bl,102
      div bl
      inc al
      mov bl,102
      mul bl
      mov si,ax
      jmp cmpnext


key   ·
   mov ax,100h
   mov ds,ax
   mov ax,si
   mov bl,102
   div bl
   mul bl
;------------------------roll back to start of record and display it
   mov si,ax
   mov cx,102
   push es
   push di
   mov ax,20h
   mov es,ax
   mov di,0

disprec    mov al,[ds:si]
   mov ah,14
   mov bh,0
   int 10h
   mov [es:di],al
   inc di
   inc si
   loop disprec
   pop di
   pop es
   jmp near exit
;-----------------------------------------display record
;   mov cx,102
;   mov di,0
;   cdisp mov al,[es:di]
;   mov ah,14
;   mov bh,0
;   int 10h
;   inc di
;   loop cdisp
;  jmp exit
npres
;---------------------------------------
        pop di
        pop si
        pop ds
        pop es

        mov BYTE al,[cursect]
        cmp al,18
        jl near incsect
        mov BYTE [cursect],1
        mov BYTE al,[curhead]
        cmp al,1
        jl near inchead
```

```
      mov BYTE [curhead],0
      mov BYTE al,[curtrack]
      inc al
      mov BYTE [curtrack],al
      jmp near next
inchead  mov BYTE [curhead],1
      jmp near next1
incsect  mov BYTE al,[cursect]
      inc al
      mov BYTE [cursect],al
next     nop
next1    mov BYTE al,[curtrack]
      mov BYTE ah,[maxtrack]
      cmp al,ah
      jl near nextsector
      mov BYTE al,[curhead]
      mov BYTE ah,[maxhead]
      cmp al,ah
      jl near nextsector
      mov BYTE al,[cursect]
      mov BYTE ah,[maxsect]
      cmp al,ah
      jle near nextsector
;-----------------------------------------------------------------
exit
mov ax,20h
mov es,ax
mov di,0
cmp BYTE [es:di],'#'
jne near exit1

mov di,83              ;INITIALISE FOR READ FILE
mov BYTE ch,[es:di]
inc di
inc di
mov BYTE dh,[es:di]
inc di
inc di
mov BYTE cl,[es:di]
mov di,93
mov BYTE ah,[es:di]
mov di,87
mov BYTE al,[es:di]
cmp ah,al
;--------------------------------------------
;-----------------FILE LOCATED IN SAME HEADS  TRACKS
```

```
jl sect
sub ah,al
inc ah
mov al,ah
mov ah,2
mov bx,ax
mov ax,2000h
mov es,ax
mov ax,bx
xor bx,bx
int 13h
jmp dic
;--------------------------------FILE LOCATED IN DIFFERENT TRACKS OR HEADS

sect mov ah,18
    sub ah,al
    inc ah
    mov al,ah
mov ah,2
mov bx,ax
mov ax,2000h
mov es,ax
mov ax,bx
xor bx,bx
int 13h
;--------------------------READ FIRST PART FULLY

mov di,0
mov cx,1024
mov ah,14
mov bh,0
disp2 mov BYTE al,[es:di]
    inc di
    int 10h
    loop disp2
xor ah,ah
int 16h

;------------------------------DISPLAY IT
mov ax,3000h   ;INITIALISE FOR THE SECOND PART OF FILE
mov es,ax
mov di,89
mov BYTE ch,[es:di]
inc di
inc di
mov BYTE dh,[es:di]
mov cl,1
inc di
inc di
mov BYTE al,[es:di]
mov ah,2
mov bx,ax
mov ax,2001h
mov es,ax
mov ax,bx
xor bx,bx
int 13h
;--------------------------------DISPLAY THE CONTENTS
mov di,0
mov ah,14
mov bh,0
disp1 mov BYTE al,[es:di]
    cmp al,255
    je outloop
    inc di
    int 10h
    jmp disp1
outloop    jmp exit1

;--------------------------------DISPLAYFIRST PART

mov ah,01
mov dx,0
int 17h


dic mov di,0
mov cx,0
disp mov BYTE al,[es:di]
    cmp al,255
    je loopout
    inc di
    inc cx
    mov ah,00
    mov dx,0
    int 17h
    jmp disp
loopout nop

mov dx,0
mov ax,cx
mov bl,80
```

```
div bl
sub bl,ah
mov cl,bl
;mov al,cl
;dispchar
pchar mov al,20h
   mov ah,00
   mov dx,0
   int 17h
   loop pchar

       pop di
       pop si
       pop ds
       pop es
       writestring message2,19,20,10
       jmp bye
exit1   pop di
       pop si
       pop ds
       pop es
       writestring message1,16,20,10
bye     xor ah,ah
       int 16h
       ;mov ax,S4c
       ;int 21h

;--------------------------------------------------------------------------
[section .data]
username db 'anon        '
message1 db 'FILE NOT PRESENT'
message2 db 'PRINT JOB COMPLETED'
[section .bss]
maxtrack resb 1
maxhead  resb 1
maxsect  resb 1
curtrack resb 1
curhead  resb 1
cursect  resb 1


;PROGRAM TO VIEW THE CONTENTS OF A FILE...

[org 0x0100]
[section .text]
%macro writestring 4      ;macro to write the string
```

```
   mov ax,%1            ;the offset value to bp thru ax
   mov bp,ax
   mov ah,13h           ;move function number to ah
   mov al,1            ;move write mode to al
   mov bh,0            ;move vdu page no to  bh
   mov bl,14           ;move attribute to bl
   mov cx,%2            ;move the stringlength to cx as parameter2
   mov dh,%3            ;move row to dh reg
   mov dl,%4            ;move col to dl
   int 10h             ;call the interrupt
%endmacro
%macro keypress 0
xor ah,ah
int 16h
%endmacro
push es
push ds
push si
push di
push es
push ds
push si
push di

        mov ax,60h
        mov es,ax
        mov di,0
        mov cx,13
        mov si,username
name    mov BYTE al,[ds:si]
        mov BYTE [es:di],al
        inc si
        inc di
        loopnz name

        mov ax,10h
        mov es,ax
        mov di,0
getkey  xor ah,ah
        int 16h
        cmp al,13
        je endofstr
        mov [es:di],al
        inc di
        mov ah,14
        mov bh,0
```

```
      int 10h
      jmp getkey

endofstr  mov ax,33
       mov cx,di
       sub ax,cx
       mov cx,ax
fillspace mov al,20h
       mov [es:di],al
       inc di
       loop fillspace
;----------------------------------------
       mov ax,12h
       mov es,ax
       mov di,0
       xor bx,bx
       mov ah,2
       mov al,1
       mov ch,0
       mov cl,1
       mov dh,1
       mov dl,0
       int 13h
       mov di,010ah
       mov BYTE al,[es:di]
       mov BYTE [maxtrack],al
       mov di,010ch
       mov BYTE al,[es:di]
       mov BYTE [maxhead],al
       mov di,010eh
       mov BYTE al,[es:di]
       mov BYTE [maxsect],al
       mov BYTE [curtrack],0
       mov BYTE [curhead],1
       mov BYTE [cursect],2
       pop di
       pop si
       pop ds
       pop es
 ;-----------------------------------------------------------------
 nextsector push es
         push ds
         push si
         push di
         mov ax,100h
         mov es,ax
```

```
        mov bx,0
        mov ah,2
        mov al,1
        mov ch,[curtrack]
        mov cl,[cursect]
        mov dh,[curhead]
        mov dl,0
        int 13h
;---------------------------------------
       mov ax,100h
       mov ds,ax
       mov ax,10h
       mov es,ax
       mov si,0
       mov di,0
cmpnext  inc si
       inc si
           mov di,0
           mov cx,33
           repe cmpsb
           cmp cx,0
           jne nequal
           ;comparing user name
           inc si
           mov ax,60h
           mov es,ax
           mov di,0
           mov cx,13
           repe cmpsb
           cmp cx,0
           je key


nequal
        cmp si,445
        jge npres
        mov ax,si
        mov bl,102
        div bl
        inc al
        mov bl,102
        mul bl
        mov si,ax
        jmp cmpnext

key
```

```
        mov ax,100h
        mov ds,ax
        mov ax,si
        mov bl,102
        div bl
        mul bl
;------------------------roll back to start of record and display it
    mov si,ax
    mov cx,102
    push es
    push di
    mov ax,20h
    mov es,ax
    mov di,0

disprec    mov al,[ds:si]
          mov ah,14
          mov bh,0
          int 10h
          mov [es:di],al
          inc di
          inc si
          loop disprec
          pop di
          pop es
         jmp near exit
;------------------------------------------display record
npres
;----------------------------------------
        pop di
        pop si
        pop ds
        pop es

        mov BYTE al,[cursect]
        cmp al,18
        jl near incsect
        mov BYTE [cursect],1
        mov BYTE al,[curhead]
        cmp al,1
        jl near inchead
        mov BYTE [curhead],0
        mov BYTE al,[curtrack]
        inc al
        mov BYTE [curtrack],al
        jmp near next
```

```
inchead  mov BYTE [curhead],1
       jmp near next1
incsect  mov BYTE al,[cursect]
       inc al
       mov BYTE [cursect],al
next    nop
next1    mov BYTE al,[curtrack]
       mov BYTE ah,[maxtrack]
       cmp al,ah
       jl near nextsector
       mov BYTE al,[curhead]
       mov BYTE ah,[maxhead]
       cmp al,ah
       jl near nextsector
       mov BYTE al,[cursect]
       mov BYTE ah,[maxsect]
       cmp al,ah
       jle near nextsector

;--------------------------------------------------------------------
exit
mov ax,20h
mov es,ax
mov di,0
cmp BYTE [es:di],'#'
jne near exit2

mov di,83                   ;INITIALISE FOR READ FILE
mov BYTE ch,[es:di]
inc di
inc di
mov BYTE dh,[es:di]
inc di
inc di
mov BYTE cl,[es:di]
mov di,93
mov BYTE ah,[es:di]
mov di,87
mov BYTE al,[es:di]
cmp ah,al
;--------------------------------------------
;-------------------FILE LOCATED IN SAME HEADS  TRACKS
jl sect
sub ah,al
inc ah
mov al,ah
mov ah,2
mov bx,ax
mov ax,2000h
mov es,ax
mov ax,bx
xor bx,bx
int 13h
jmp dic
;--------------------------------FILE LOCATED IN DIFFERENT TRACKS OR HEADS

sect mov ah,18
    sub ah,al
    inc ah
    mov al,ah
mov ah,2
mov bx,ax
mov ax,2000h
mov es,ax
mov ax,bx
xor bx,bx
int 13h
;---------------------------READ FIRST PART FULLY

mov di,0
mov cx,1024
mov ah,14
mov bh,0
disp2 mov BYTE al,[es:di]
    inc di
    int 10h
    loop disp2
xor ah,ah
int 16h

;------------------------------DISPLAY IT
mov ax,20h   ;INITIALISE FOR THE SECOND PART OF FILE
mov es,ax
mov di,89
mov BYTE ch,[es:di]
inc di
inc di
mov BYTE dh,[es:di]
mov cl,1
inc di
inc di
mov BYTE al,[es:di]
```

```
        ;mov ax,$4c
        ;int 21h
        retf

;-------------------------------------------------------------------
[section .data]
username db 'anon          '
nofile db 'FILE NOT FOUND...'
success db 'PRESS ANY KEY TO CONTINUE...'

[section .bss]
maxtrack resb 1
maxhead  resb 1
maxsect  resb 1
curtrack resb 1
curhead  resb 1
cursect  resb 1


;THIS IS THE DISKVIEWER UTILITY FOR THE CREAGX OPERATING SYSTEM
[ORG 0X0100]
[SECTION .text]
jmp startofdisp
%macro exit 0
        mov ax,$4c
        int 21h
%endmacro
%macro   clrscr 0
        mov cx,2000
        label mov ah,14
        mov al,' '
        xor bh,bh
        int 10h
        loop label
        setcursor 0,0
%endmacro
%macro setcursor 2
        mov ah,2
        mov bh,0
        mov dh,%1
        mov dl,%2
        int 10h
%endmacro
%macro dispchar 0
        mov ah,14
        xor bh,bh
```

```
      int 10h
%endmacro
%macro keypress 0
      xor ah,ah
      int 16h
%endmacro
%macro print_at 5        ;macro to write the string
   mov ax,%1             ;move the offset value to bp thru ax
   mov bp,ax
   mov ah,13h             ;move function number to ah
   mov al,1              ;move write mode to al
   mov bh,0              ;move vdu page no to  bh
   mov bl,%5             ;move attribute to bl
   mov cx,%2             ;move the stringlength to cx as parameter2
   mov dh,%3             ;move row to dh reg
   mov dl,%4             ;move col to dl reg
   int 10h               ;call the interrupt
%endmacro

%macro incline 0
   push cx
   mov ah,3
   mov bh,0
   int 10h
   add dh,1
   mov dl,10
   mov ah,2
   mov bh,0
   int 10h
   pop cx
%endmacro

procascii2bin:
       push ax
       shr al,4
       cmp al,9
       jle near ndeci
       sub al,9
       add al,64
       dispchar
       jmp near next
ndeci   add al,48
       dispchar
next    pop ax
       and al,15
       cmp al,9
```

```
        dec al
        mov [unit],al
        cmp al,0

        jg near dispchs
        mov al,18
        mov [unit],al
        mov al,[side]
        cmp al,1
        je near prevhead
        mov al,1
        mov [side],al
        mov al,[track]
        dec al
        mov [track],al
        cmp al,0
        jg near dispchs
        mov al,79
        mov [track],al
        jmp near dispchs
prevhead  dec al
        mov [side],al


dispchs   setcursor 2,10
        mov al,[track]
        call procascii2bin
        setcursor 2,42
        mov al,[side]
        call procascii2bin
        setcursor 2,77
        mov al,[unit]
        call procascii2bin
        jmp near chschange
over      exit
[SECTION .data]
wmess db 'WELCOME TO CREAGX DISK VIEWER'
cylinder db 'CYLINDER'
head     db  'HEAD'
sector  db 'SECTOR'
[SECTION .bss]
track  resb 1
side  resb 1
unit  resb 1
```