OPERATING SYSTEM

CSE 301 L

Project Report on

“CREATING AN OPERATIN SYSTEM ”

Submitted by:

TALLURI NAGESWARA RAO AP21110011511

CHAVALA AHALYA AP21110011512

Under the guidance of

Dr. ASHOK KUMAR PRADHAN

INDEX

INTRODUCTION :

OPERATING SYSTEM :

- It acts as an intermediary between the user of computer and it's hardware.
- Operating system is a program running at all times on the computer and acts interface between user and hardware

The purpose of an operating system is to provide an environment in which use to execute programs conveniently and efficiently. It provides memory, processors(instances of its), devices to all process by the help of CPU scheduling algorithms.

BASIC FEATURES :

- Convenience :

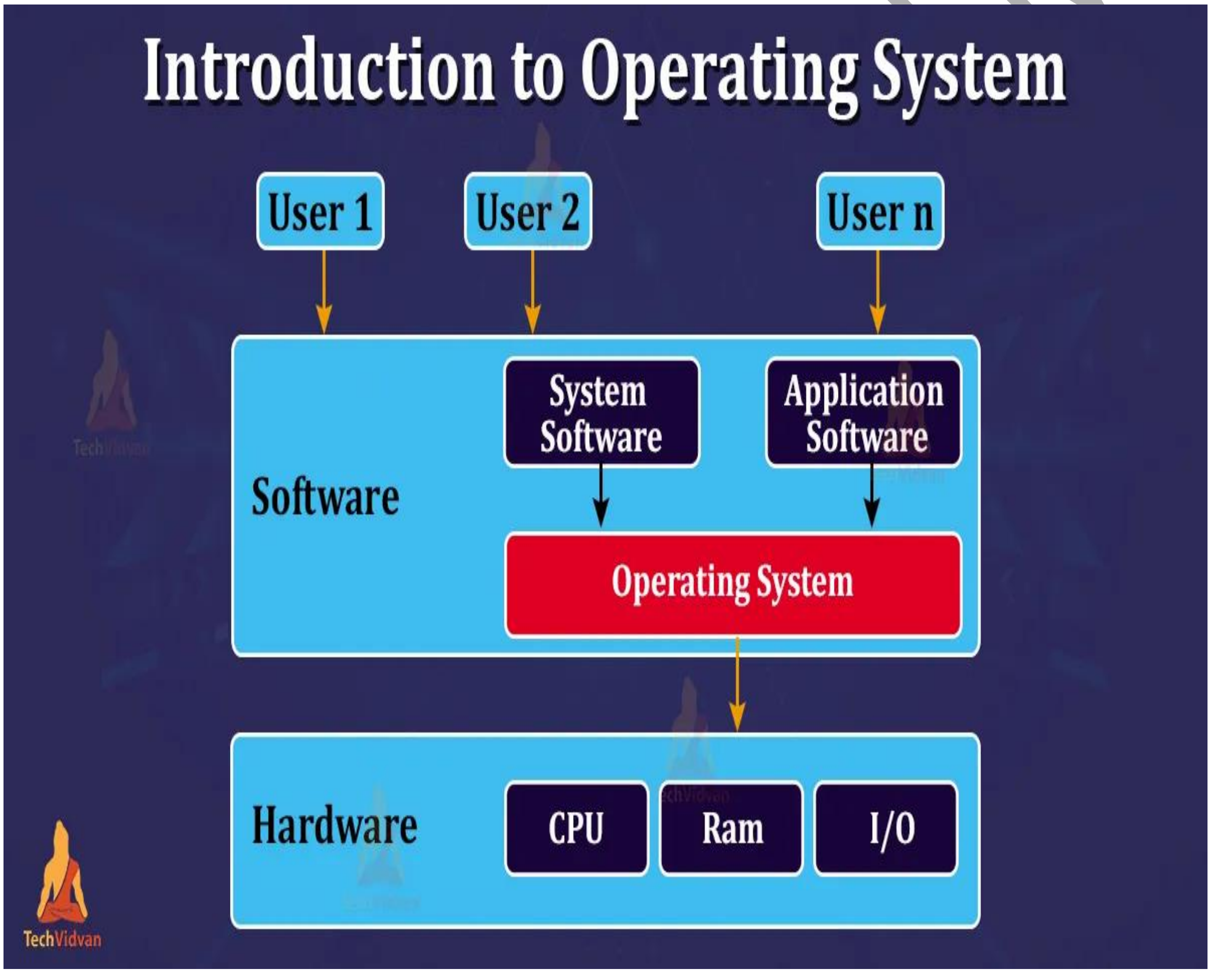
  Operating system makes computer to more convenient to use by providing processors and memory to the process as they required.
- Efficiency :

  Operating system does not allow the CPU to sit idle because of to improve the utilization of CPU then it leads to the efficiency use of computer.

➢ Throughput:

Operating system is constructed to provide it's maximum by providing resources.

Number of tasks per unit time.

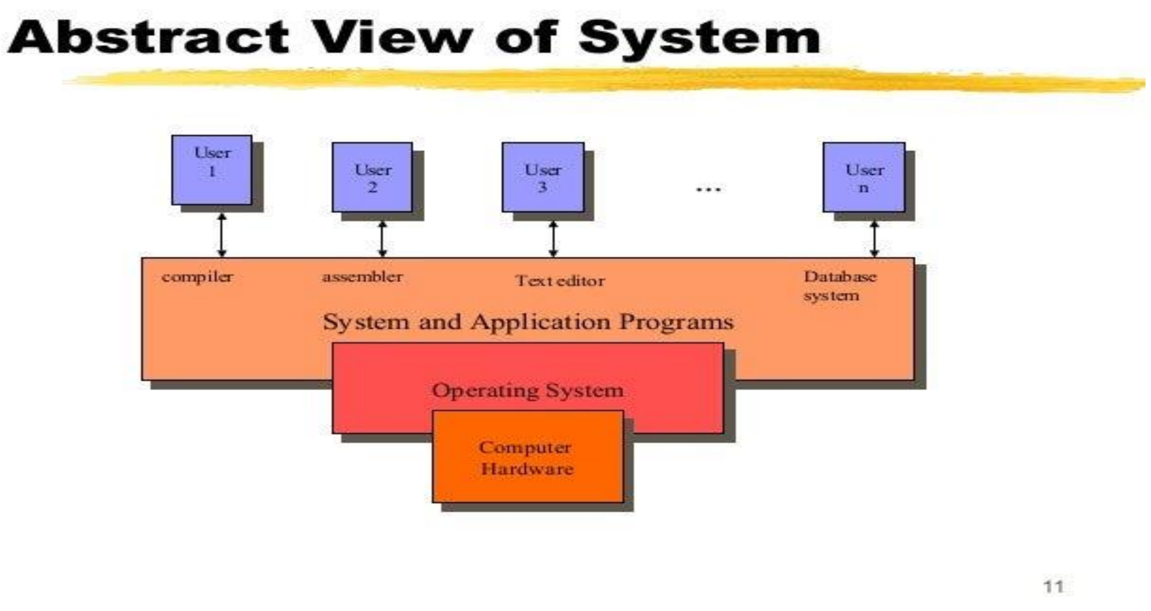


ABSTRACT :

In this project we are discussing about “creating an operating system” and it’s built in functions. In this project we are developing 64 bit operating system using ubantu resources and it’s applications.

In this project our operating system name is “SRMOS” which is developed using bochs software, gcc , nasm assembler.

WSL (windows subsystem linux) is play key role to create operating system when ever we are using windows and in other operating system it does not need but we need to install appropriate software to support the operating system.



SYSTEM REQUIREMENTS & IMPLIMENTATION :

LINUX :-

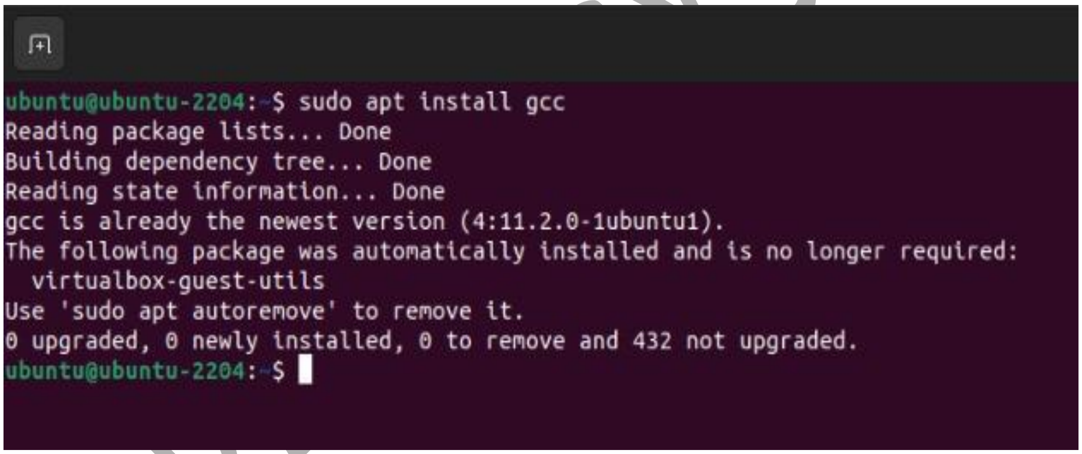
In this project we are using ubantu terminal, So we need to learn commands and requirements regarding to the ubantu terminal.

C and C++ programs :

- **Installing :**

  For installing gcc compiler :

  "sudo apt install gcc"

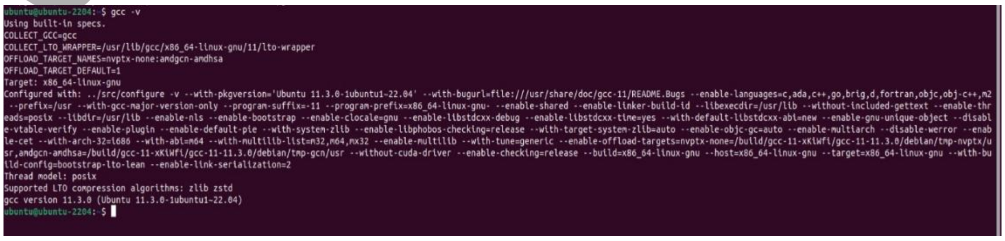


```
ubuntu@ubuntu-2204:~$ sudo apt install gcc
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
gcc is already the newest version (4:11.2.0-1ubuntu1).
The following package was automatically installed and is no longer required:
  virtualbox-guest-utils
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 432 not upgraded.
ubuntu@ubuntu-2204:~$ 
```

- **Verify version :**

  To find our gcc vrsion :

  "gcc -v"



```
ubuntu@ubuntu-2204:~$ gcc -v
Using built-in specs.
COLLECT_GCC=gcc
COLLECT_LTO_WRAPPER=/usr/lib/gcc/x86_64-linux-gnu/11/lto-wrapper
OFFLOAD_TARGET_NAMES=nvptx-none:amdgcn-amdhsa
OFFLOAD_TARGET_DEFAULT=1
Target: x86_64-linux-gnu
Configured with: ../src/configure -v --with-pkgversion='Ubuntu 11.3.0-1ubuntu1~22.04' --with-bugurl=file:///usr/share/doc/gcc-11/README.Bugs --enable-languages=c,ada,c++,go,brig,d,fortran,objc,obj-c++,m2
 --prefix=/usr --with-gcc-major-version-only --program-suffix=-11 --program-prefix=x86_64-linux-gnu- --enable-shared --enable-linker-build-id --libexecdir=/usr/lib --without-included-gettext --enable-thr
eads=posix --libdir=/usr/lib --enable-nls --enable-bootstrap --enable-clocale=gnu --enable-libstdcxx-debug --enable-libstdcxx-time=yes --with-default-libstdcxx-abi=new --enable-gnu-unique-object --disabl
e-vtable-verify --enable-plugin --enable-default-pie --with-system-zlib --enable-libphobos-checking=release --with-target-system-zlib=auto --enable-objc-gc=auto --enable-multiarch --disable-werror --enab
le-cet --with-arch-32=i686 --with-abi=m64 --with-multilib-list=m32,m64,mx32 --enable-multilib --with-tune=generic --enable-offload-targets=nvptx-none=/build/gcc-11-xKiWfi/gcc-11-11.3.0/debian/tmp-nvptx/u
sr,amdgcn-amdhsa=/build/gcc-11-xKiWfi/gcc-11-11.3.0/debian/tmp-gcn/usr --without-cuda-driver --enable-checking=release --build=x86_64-linux-gnu --host=x86_64-linux-gnu --target=x86_64-linux-gnu --with-bu
ild-config=bootstrap-lto-lean --enable-link-serialization=2
Thread model: posix
Supported LTO compression algorithms: zlib zstd
gcc version 11.3.0 (Ubuntu 11.3.0-1ubuntu1~22.04)
ubuntu@ubuntu-2204:~$ 
```

It is 64 bit compiler

NASM(Assembly Language) :

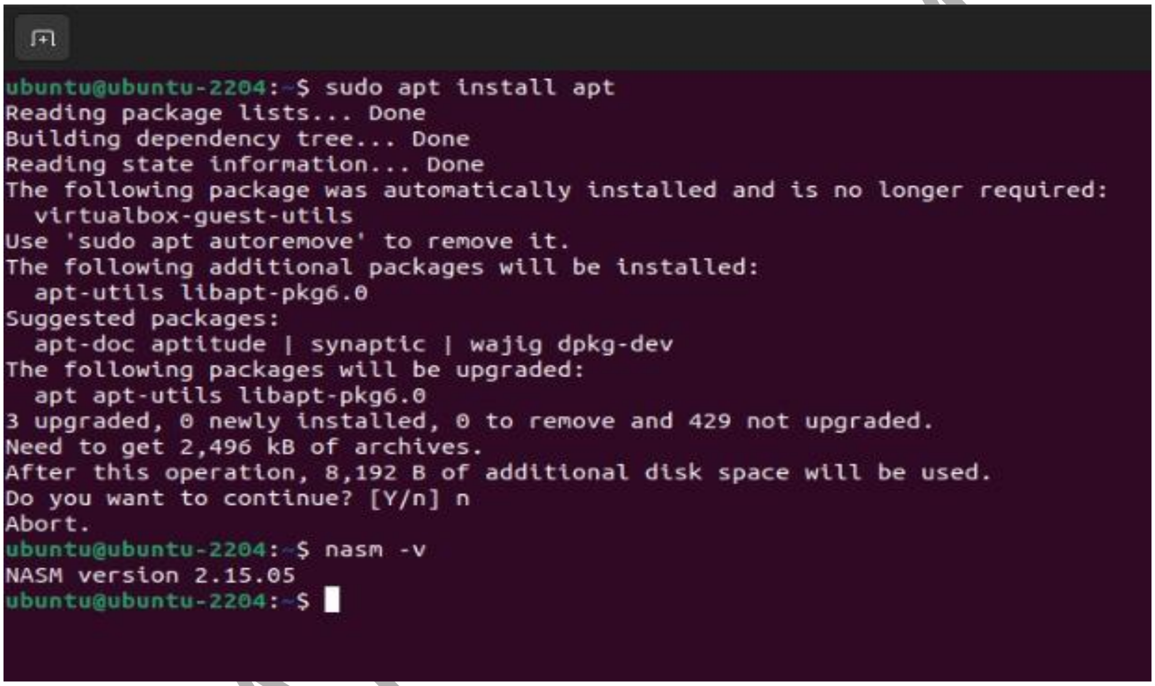
➢ Installing

For installing NASM compiler

“sudo apt install nasm”

For verifying version :

“nasm -v”



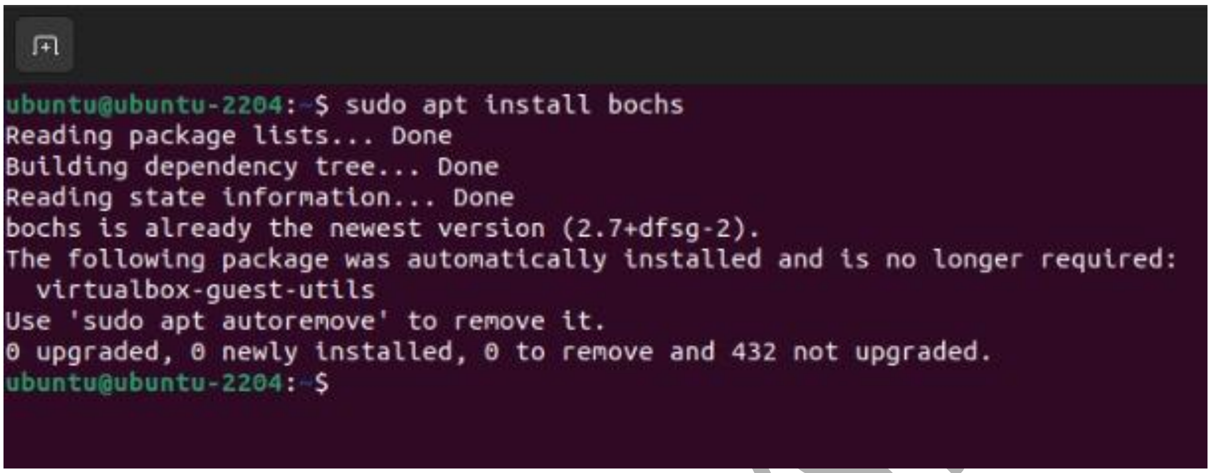
```
ubuntu@ubuntu-2204:~$ sudo apt install apt
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following package was automatically installed and is no longer required:
  virtualbox-guest-utils
Use 'sudo apt autoremove' to remove it.
The following additional packages will be installed:
  apt-utils libapt-pkg6.0
Suggested packages:
  apt-doc aptitude | synaptic | wajig dpkg-dev
The following packages will be upgraded:
  apt apt-utils libapt-pkg6.0
3 upgraded, 0 newly installed, 0 to remove and 429 not upgraded.
Need to get 2,496 kB of archives.
After this operation, 8,192 B of additional disk space will be used.
Do you want to continue? [Y/n] n
Abort.
ubuntu@ubuntu-2204:~$ nasm -v
NASM version 2.15.05
ubuntu@ubuntu-2204:~$ 
```

You need to proceed with by clicking ‘y’ during installation because it will provide some useful extensions for nasm compiler .

INSTALLING BOCHS SOFTWARE :

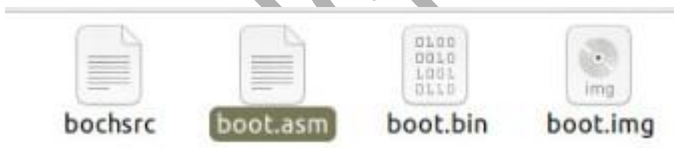
It is software which uses to emulate assembly language(low level programming language) to create operating system. It is difficult to create operating system using high level programming language like C, C++, python, Java…… etc because these language can not interact with registers which are present in the CPU, So level programming language is suitable to create operating system.

Bochs software will helps us to develop operating system. We will further study about bochs software in detail.



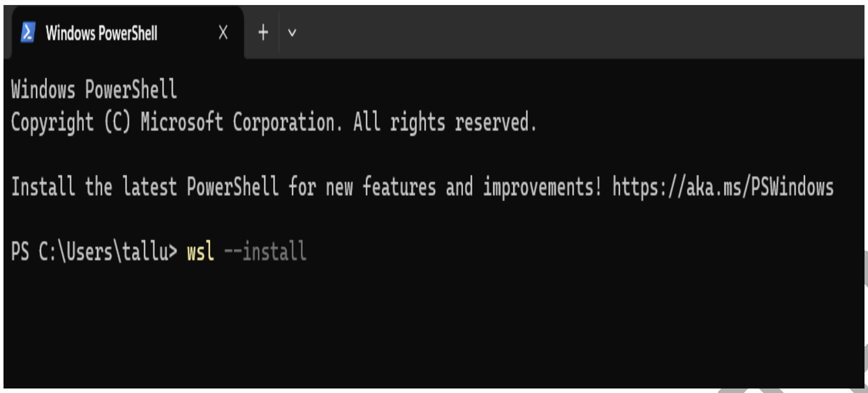
```
ubuntu@ubuntu-2204:~$ sudo apt install bochs
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
bochs is already the newest version (2.7+dfsg-2).
The following package was automatically installed and is no longer required:
  virtualbox-guest-utils
Use 'sudo apt autoremove' to remove it.
0 upgraded, 0 newly installed, 0 to remove and 432 not upgraded.
ubuntu@ubuntu-2204:~$
```

Related folders



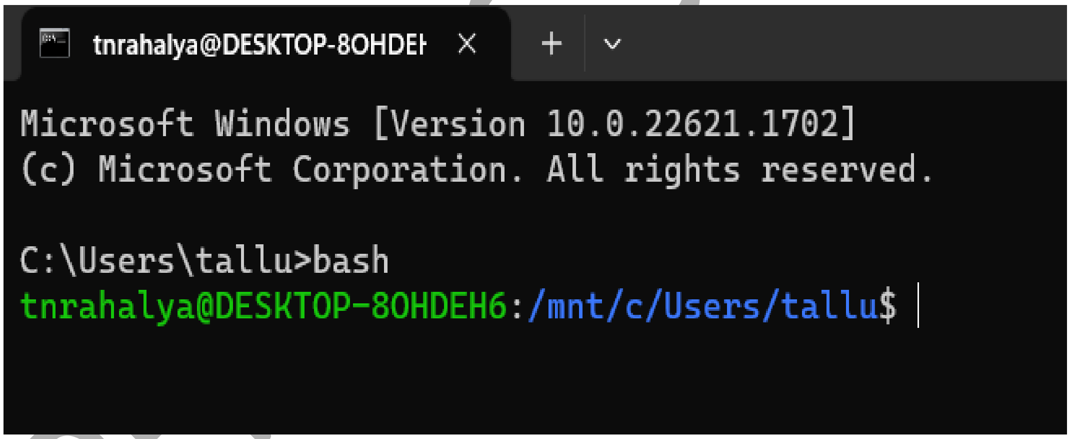
WINDOWS :-

- Install ubuntu using windows power shell



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\tallu> wsl --install
```

- Create username and password for your ubuntu operating system
- Install bochs software



```
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\tallu>bash
tnrahalya@DESKTOP-8OHDEH6:/mnt/c/Users/tallu$
```

- Set floopy disc for the booting to run the program

ASSEMBLY LANGUAGE

An assembly language is a type of low-level programming language which is intended to communicate directly with a computer's hardware. It is different to high level programming languages (convert into machine level language) because it consists of binary and hexadecimal characters and also readable by humans.

Syntax :

The rules are defined for writing assembly language is very simple and easy to understand.

Label :

The labels present in the assembly language are related to the function in high level programming language.

It ends with colons (:)

It does not have any return values and parameters like functions in high level programming language.

Declaration :

Label1:

……………

……………..

Arguments

……………..

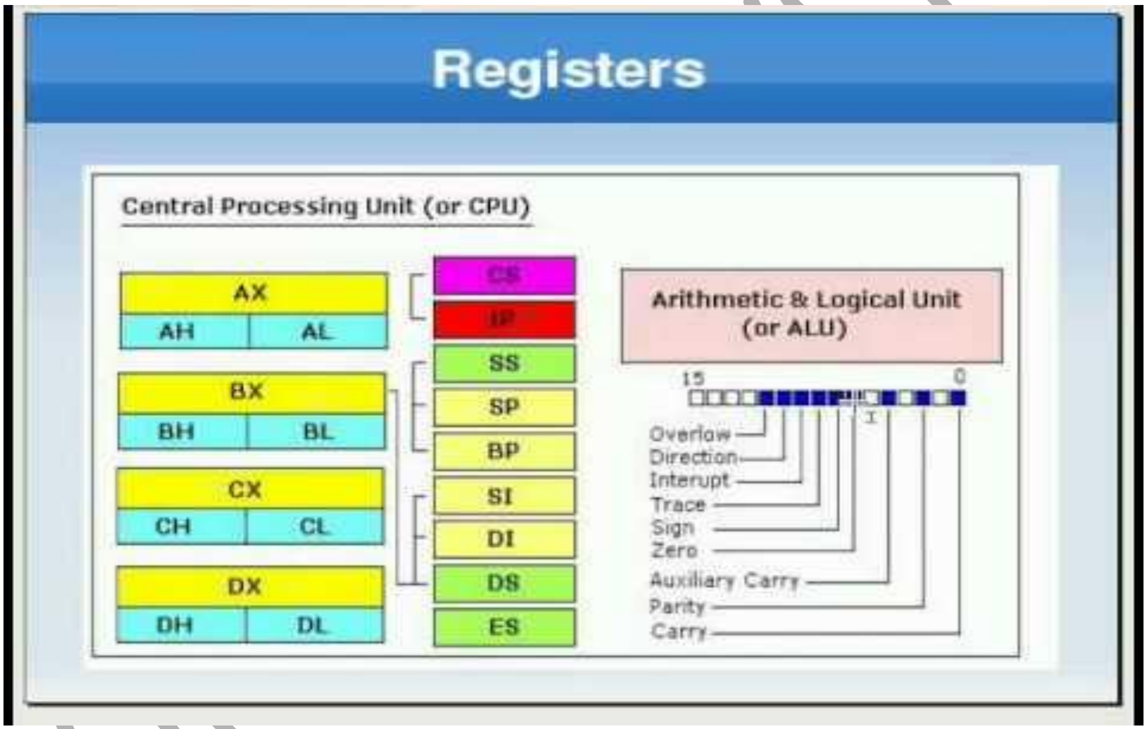
Directives :

It contains some predefined functions like as high level programming languges.

EX:

#include<emu8086.in>

REGISTERS :



KEYWORDS :

	CMPSB				MOV		
AAA	CMPSW	JAE	JNBE	JPO	MOVSB	RCR	SCASB
AAD	CWD	JB	JNC	JS	MOVSW	REP	SCASW
AAM	DAA	JBE	JNE	JZ	MUL	REPE	SHL
AAS	DAS	JC	JNG	LAHF	NEG	REPNE	SHR
ADC	DEC	JCXZ	JNGE	LDS	NOP	REPNZ	STC
ADD	DIV	JE	JNL	LEA	NOT	REPZ	STD
AND	HLT	JG	JNLE	LES	OR	RET	STI
CALL	IDIV	JGE	JNO	LODSB	OUT	RETF	STOSB
CBW	IMUL	JL	JNP	LODSW	POP	ROL	STOSW
CLC	IN	JLE	JNS	LOOP	POPA	ROR	SUB
CLD	INC	JMP	JNZ	LOOPE	POPF	SAHF	TEST
CLI	INT	JNA	JO	LOOPNE	PUSH	SAL	XCHG
CMC	INTO	JNAE	JP	LOOPNZ	PUSHA	SAR	XLATB
CMP	IRETJA	JNB	JPE	LOOPZ	PUSHF	SBB	XOR
					RCL		

The above keywords are used to work with assembly language. Find the function of each keyword in either youtube or google or in some other references.

In above keywords interrupt key words are not used :

RET,IRET are used to create interrupts.

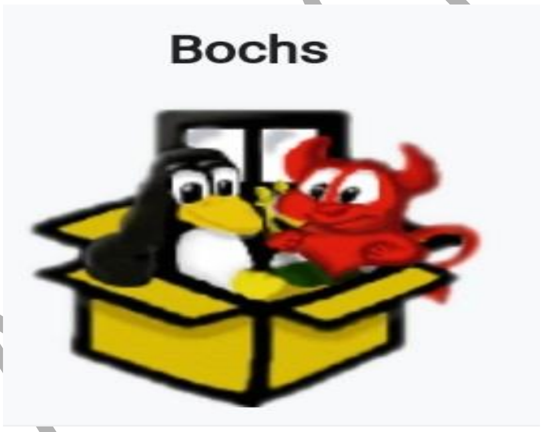
BOCHS SOFTWARE

It is software which uses to emulate assembly language(low level programming language) to create operating system.

It is free software under the GNU lesser general public services.

The operating system which are using these software to run their programs are :

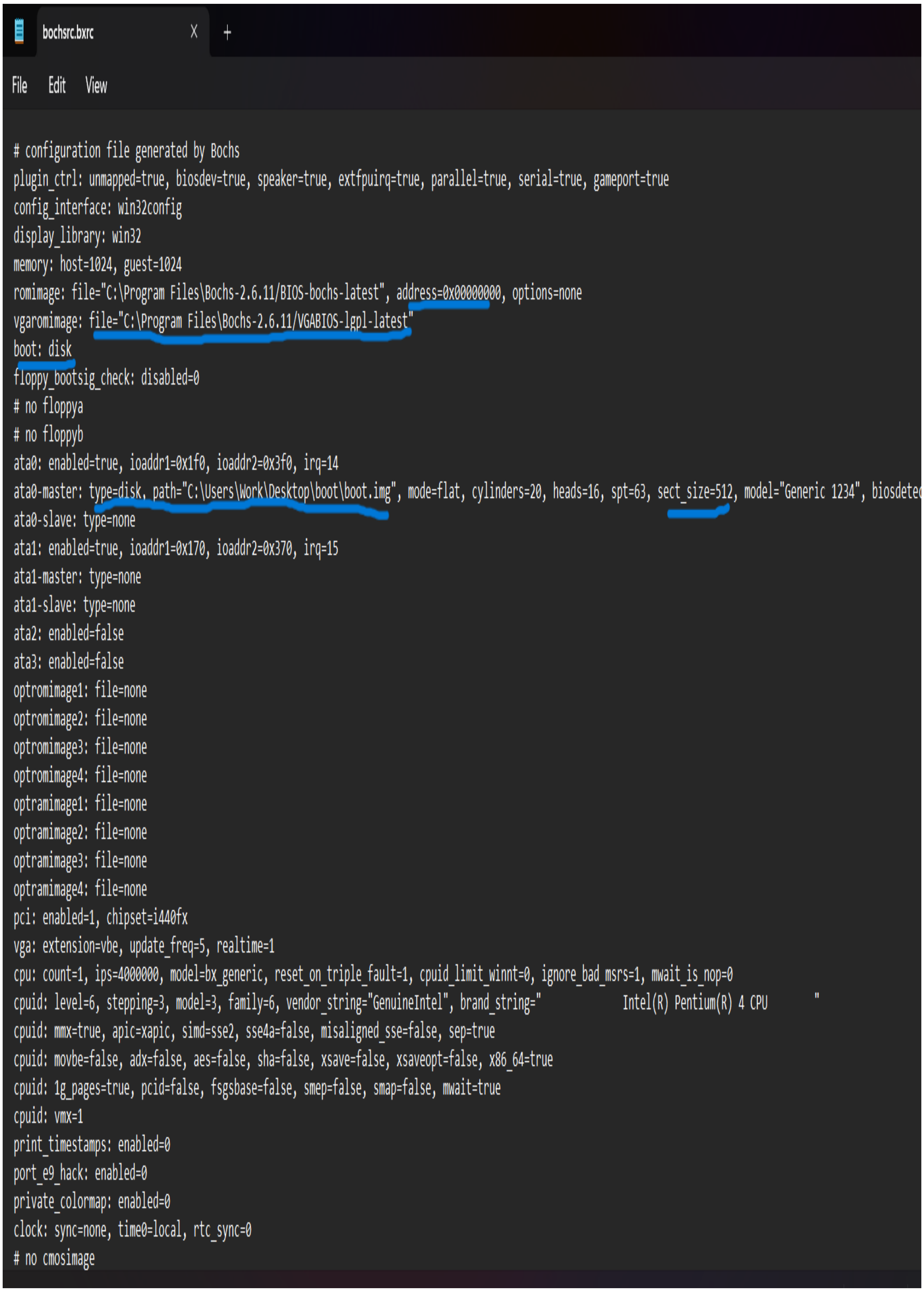
1. DOS
2. Several versions of Windows, linux, BSD's,
3. Android



Bochs is mostly used for operating system development(when an emulated operating system crashes, It doesn't crashes the host operating system, So the emulated OS can be debugged) and to run other guest operating systems.

It consists CD drives and floppy drives.

Configuration of bochs software with boot programming and memory setting with paths :



```
# configuration file generated by Bochs
plugin_ctrl: unmapped=true, biosdev=true, speaker=true, extfpuirq=true, parallel=true, serial=true, gameport=true
config_interface: win32config
display_library: win32
memory: host=1024, guest=1024
romimage: file="C:\Program Files\Bochs-2.6.11/BIOS-bochs-latest", address=0x00000000, options=none
vgaromimage: file="C:\Program Files\Bochs-2.6.11/VGABIOS-lgpl-latest"
boot: disk
floppy_bootsig_check: disabled=0
# no floppya
# no floppyb
ata0: enabled=true, ioaddr1=0x1f0, ioaddr2=0x3f0, irq=14
ata0-master: type=disk, path="C:\Users\Work\Desktop\boot\boot.img", mode=flat, cylinders=20, heads=16, spt=63, sect_size=512, model="Generic 1234", biosdetec
ata0-slave: type=none
ata1: enabled=true, ioaddr1=0x170, ioaddr2=0x370, irq=15
ata1-master: type=none
ata1-slave: type=none
ata2: enabled=false
ata3: enabled=false
optromimage1: file=none
optromimage2: file=none
optromimage3: file=none
optromimage4: file=none
optramimage1: file=none
optramimage2: file=none
optramimage3: file=none
optramimage4: file=none
pci: enabled=1, chipset=i440fx
vga: extension=vbe, update_freq=5, realtime=1
cpu: count=1, ips=4000000, model=bx_generic, reset_on_triple_fault=1, cpuid_limit_winnt=0, ignore_bad_msrs=1, mwait_is_nop=0
cpuid: level=6, stepping=3, model=3, family=6, vendor_string="GenuineIntel", brand_string="              Intel(R) Pentium(R) 4 CPU        "
cpuid: mmx=true, apic=xapic, simd=sse2, sse4a=false, misaligned_sse=false, sep=true
cpuid: movbe=false, adx=false, aes=false, sha=false, xsave=false, xsaveopt=false, x86_64=true
cpuid: 1g_pages=true, pcid=false, fsgsbase=false, smep=false, smap=false, mwait=true
cpuid: vmx=1
print_timestamps: enabled=0
port_e9_hack: enabled=0
private_colormap: enabled=0
clock: sync=none, time0=local, rtc_sync=0
# no cmosimage
```

PROCESS OF EXCUTION :

Creating program file with require arguments

↓ Compile

Create a bin file (binary file)

↓

Loaded into boot.image file

↓

Change the requirements and load the operating system

↓

Excute the boch file and run the operating system

COMMANDS FOR EXCUTION :

CREATING BIN FILE :

```
“  nasm -f filename.bin”
```

The above command will create bin file of assembly language will load into image file.

EXCUTION OF FILE :

```
“ nasm -o filename.asm “
```

The above state is used to run the assembly language file and to help in creating bin file.

LODING OF IMAGE FILE :

```
“   dd if=filename.bin of=filename.img   “
```

The above command is used search the bin file and load that file into image file and change the execution of operating system and run as according to our commands

EXCUTING SHELL SCRIPT FILES :

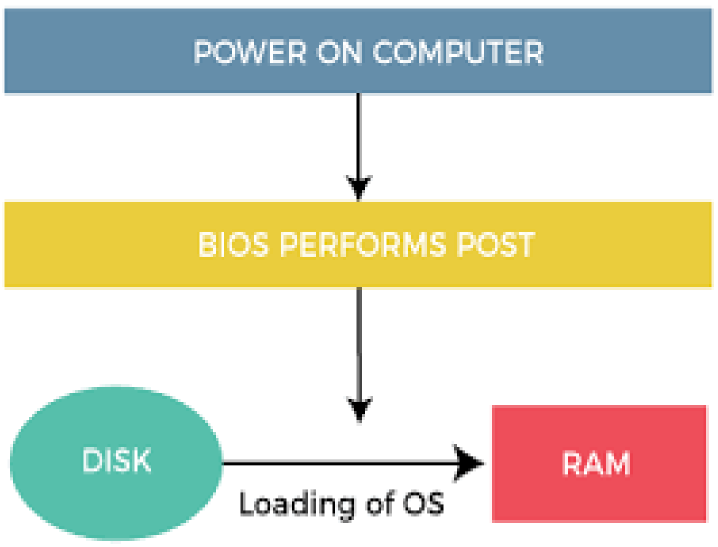
```
“   bash filename.sh  “
```

The programs which are hold our commands and there are saved in bash shell to run all files at a time.

The above command is used to execute shell script file and then files in the script file will execute according to their order.

BOOTING

Booting is program sequence which starts the operating system when it is turned on. Booting consists set of instruction which are ready to perform when computer is switch on. Every computer has booting program.



Boot loader :

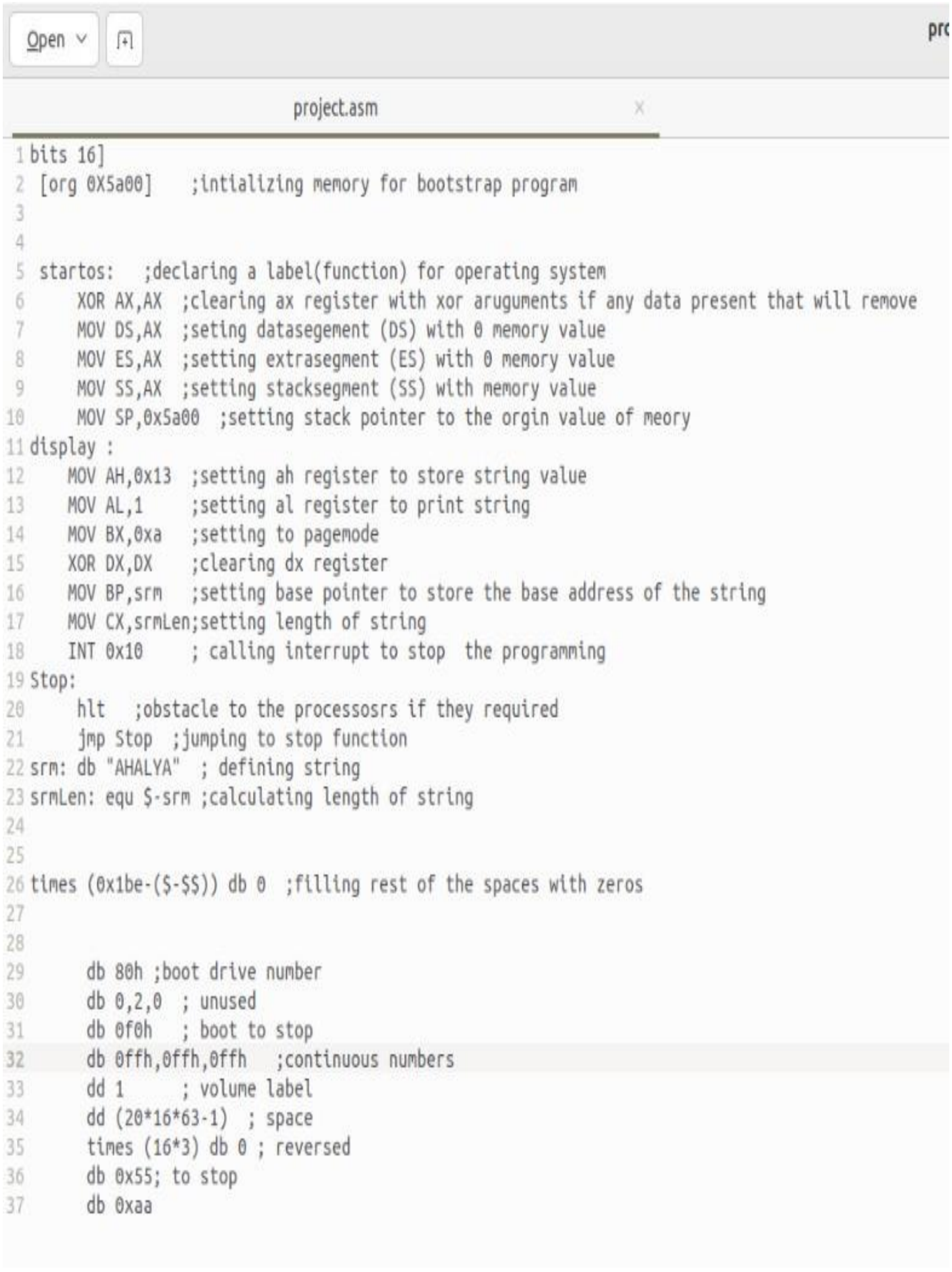
The program which is initialize the operating system to start it process is call bootloader program or bootstrap program.

Boot Devices :

The devices which are responsible for initializing operating system are called boot devices. They are

1. CD Drive
2. HardDisk Drive
3. Network

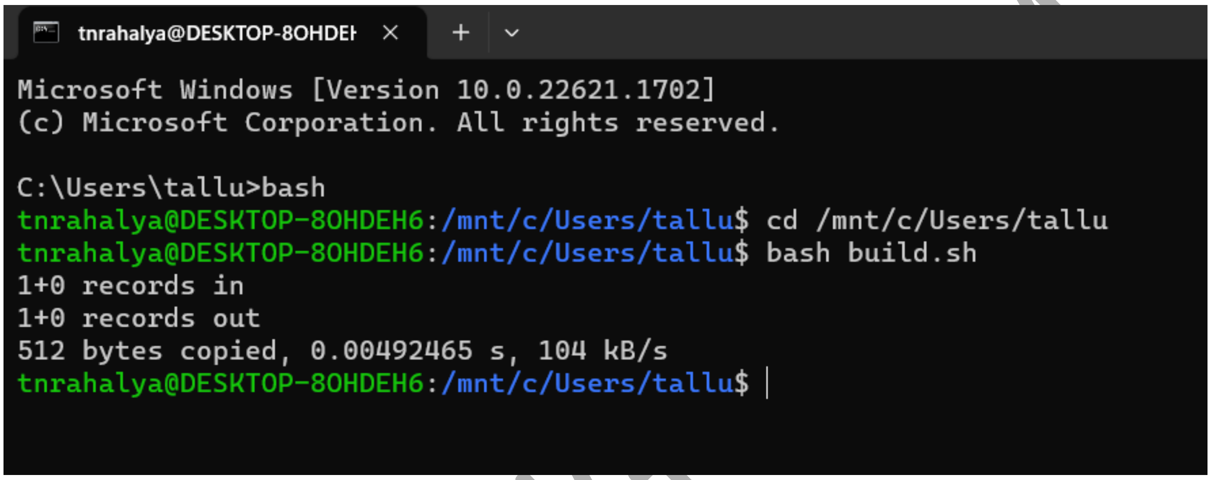
Boot program for our os :



```
bits 16]
 [org 0X5a00]    ;intializing memory for bootstrap program


 startos:   ;declaring a label(function) for operating system
     XOR AX,AX  ;clearing ax register with xor aruguments if any data present that will remove
     MOV DS,AX  ;seting datasegement (DS) with 0 memory value
     MOV ES,AX  ;setting extrasegment (ES) with 0 memory value
     MOV SS,AX  ;setting stacksegment (SS) with memory value
     MOV SP,0x5a00  ;setting stack pointer to the orgin value of meory
display :
    MOV AH,0x13  ;setting ah register to store string value
    MOV AL,1     ;setting al register to print string
    MOV BX,0xa   ;setting to pagemode
    XOR DX,DX    ;clearing dx register
    MOV BP,srm   ;setting base pointer to store the base address of the string
    MOV CX,srmLen;setting length of string
    INT 0x10     ; calling interrupt to stop  the programming
Stop:
     hlt   ;obstacle to the processosrs if they required
     jmp Stop  ;jumping to stop function
srm: db "AHALYA"  ; defining string
srmLen: equ $-srm ;calculating length of string


times (0x1be-($-$$)) db 0  ;filling rest of the spaces with zeros


      db 80h ;boot drive number
      db 0,2,0  ; unused
      db 0f0h   ; boot to stop
      db 0ffh,0ffh,0ffh   ;continuous numbers
      dd 1      ; volume label
      dd (20*16*63-1)  ; space
      times (16*3) db 0 ; reversed
      db 0x55; to stop
      db 0xaa
```

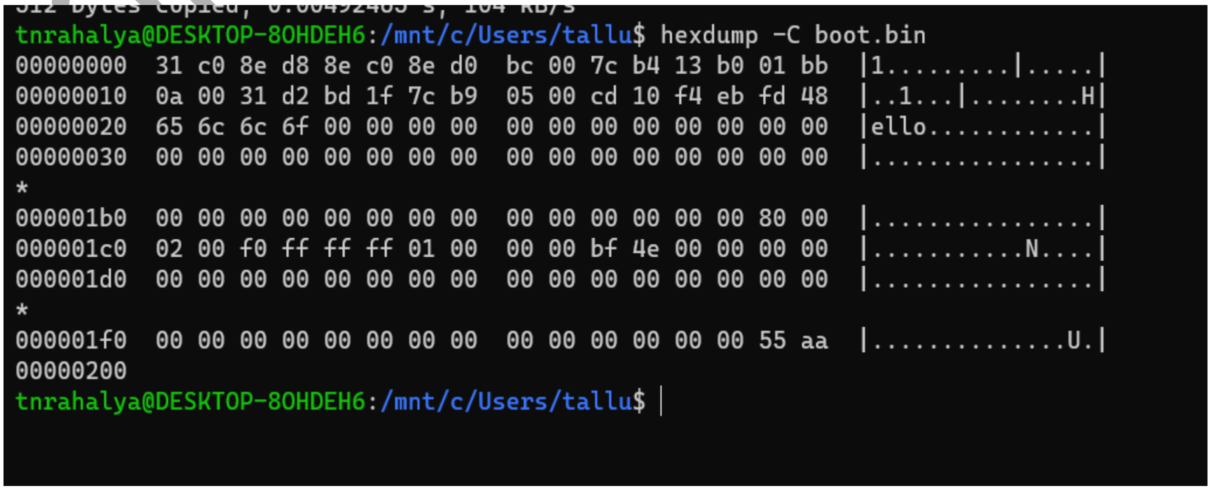
```
[bits 16]

 [org 0X5a00]    ;intializing memory for bootstrap program

  startos:   ;declaring a label(function) for operating system

     XOR AX,AX  ;clearing ax register with xor aruguments if any data present that will remove

     MOV DS,AX  ;seting datasegement (DS) with 0 memory value

     MOV ES,AX  ;setting extrasegment (ES) with 0 memory value

     MOV SS,AX  ;setting stacksegment (SS) with memory value

     MOV SP,0x5a00  ;setting stack pointer to the orgin value of meory

display :

    MOV AH,0x13  ;setting ah register to store string value

    MOV AL,1     ;setting al register to print string

    MOV BX,0xa   ;setting to pagemode

    XOR DX,DX    ;clearing dx register

    MOV BP,srm   ;setting base pointer to store the base address of the string

    MOV CX,srmLen;setting length of string

    INT 0x10     ; calling interrupt to stop  the programming

Stop:

     hlt   ;obstacle to the processosrs if they required

     jmp Stop  ;jumping to stop function

srm: db "HELLO"  ; defining string

srmLen: equ $-srm ;calculating length of string

times (0x1be-($-$$)) db 0  ;filling rest of the spaces with z

      db 80h ;boot drive number

      db 0,2,0  ; unused

      db 0f0h   ; boot to stop

      db offh,offh,offh   ;continuous numbers
```

```
dd 1      ; volume label

dd (20*16*63-1)  ; space

times (16*3) db 0 ; reversed

db 0x55; to stop

db 0xaa
```



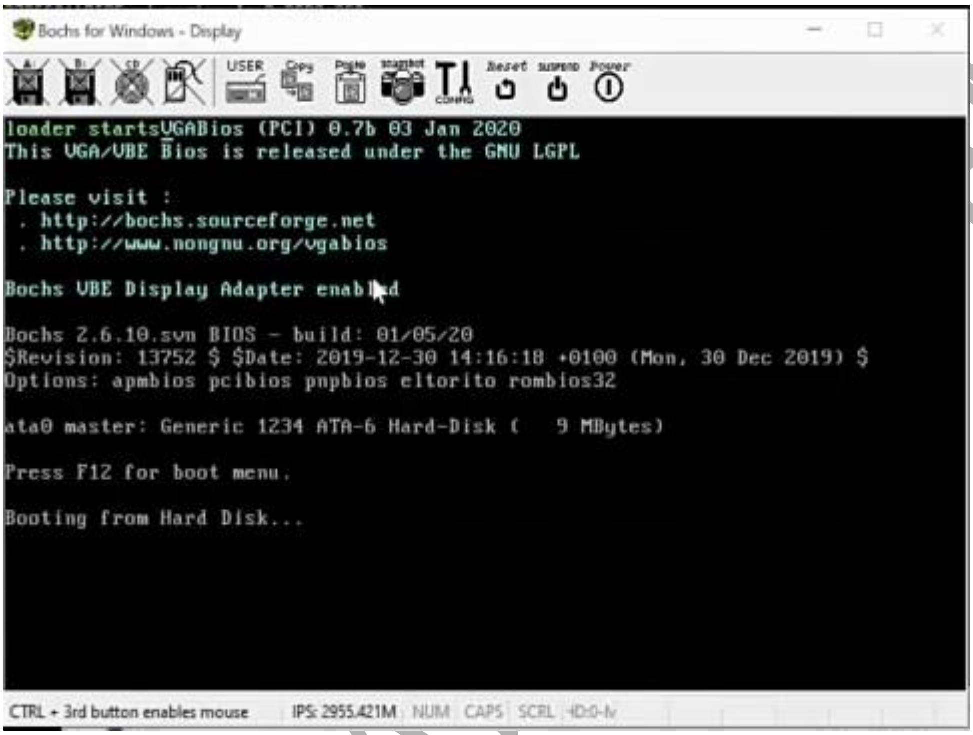
```
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\tallu>bash
tnrahalya@DESKTOP-8OHDEH6:/mnt/c/Users/tallu$ cd /mnt/c/Users/tallu
tnrahalya@DESKTOP-8OHDEH6:/mnt/c/Users/tallu$ bash build.sh
1+0 records in
1+0 records out
512 bytes copied, 0.00492465 s, 104 kB/s
tnrahalya@DESKTOP-8OHDEH6:/mnt/c/Users/tallu$
```

Booting has been completed let see the structure of the memory and CHG structure :



```
tnrahalya@DESKTOP-8OHDEH6:/mnt/c/Users/tallu$ hexdump -C boot.bin
00000000  31 c0 8e d8 8e c0 8e d0  bc 00 7c b4 13 b0 01 bb  |1.........|.....|
00000010  0a 00 31 d2 bd 1f 7c b9  05 00 cd 10 f4 eb fd 48  |..1...|........H|
00000020  65 6c 6c 6f 00 00 00 00  00 00 00 00 00 00 00 00  |ello............|
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000001b0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 80 00  |................|
000001c0  02 00 f0 ff ff ff 01 00  00 00 bf 4e 00 00 00 00  |...........N....|
000001d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
*
000001f0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 55 aa  |..............U.|
00000200
tnrahalya@DESKTOP-8OHDEH6:/mnt/c/Users/tallu$
```

The above picture is memory structure of booting of memory size 1024. The 0 values are empty spaces which are ready to store the access the proceess loading to RAM. The remaining spaces are spaces filled with memory utilized by bootstrap program



```
loader startsVGABios (PCI) 0.7b 03 Jan 2020
This VGA/VBE Bios is released under the GNU LGPL

Please visit :
 . http://bochs.sourceforge.net
 . http://www.nongnu.org/vgabios

Bochs VBE Display Adapter enabled

Bochs 2.6.10.svn BIOS - build: 01/05/20
$Revision: 13752 $ $Date: 2019-12-30 14:16:18 +0100 (Mon, 30 Dec 2019) $
Options: apmbios pcibios pnpbios eltorito rombios32

ata0 master: Generic 1234 ATA-6 Hard-Disk (   9 MBytes)

Press F12 for boot menu.

Booting from Hard Disk...
```

LOADING

Loading is program which allocate useable data to the RAM memory to run operating system.

$$\begin{tikzcd}\text{Secondary memory} \arrow[r, dashed, "\boxed{\text{Loader}}"] & \text{Main memory}\end{tikzcd}$$

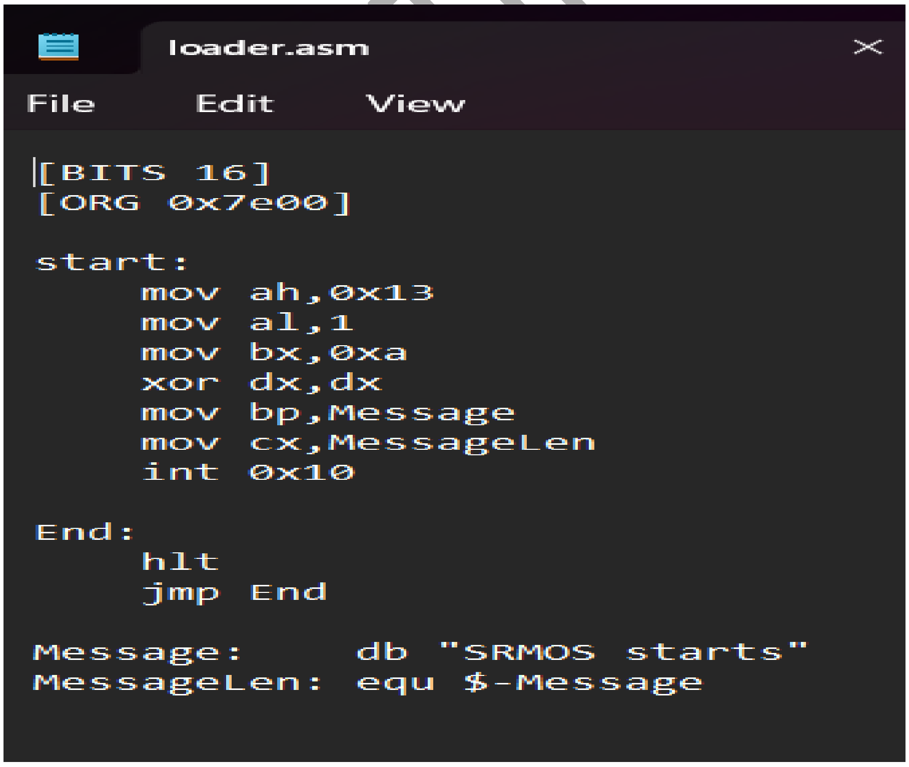
It is also known as bootstrap loader or boot loader.

It is two types

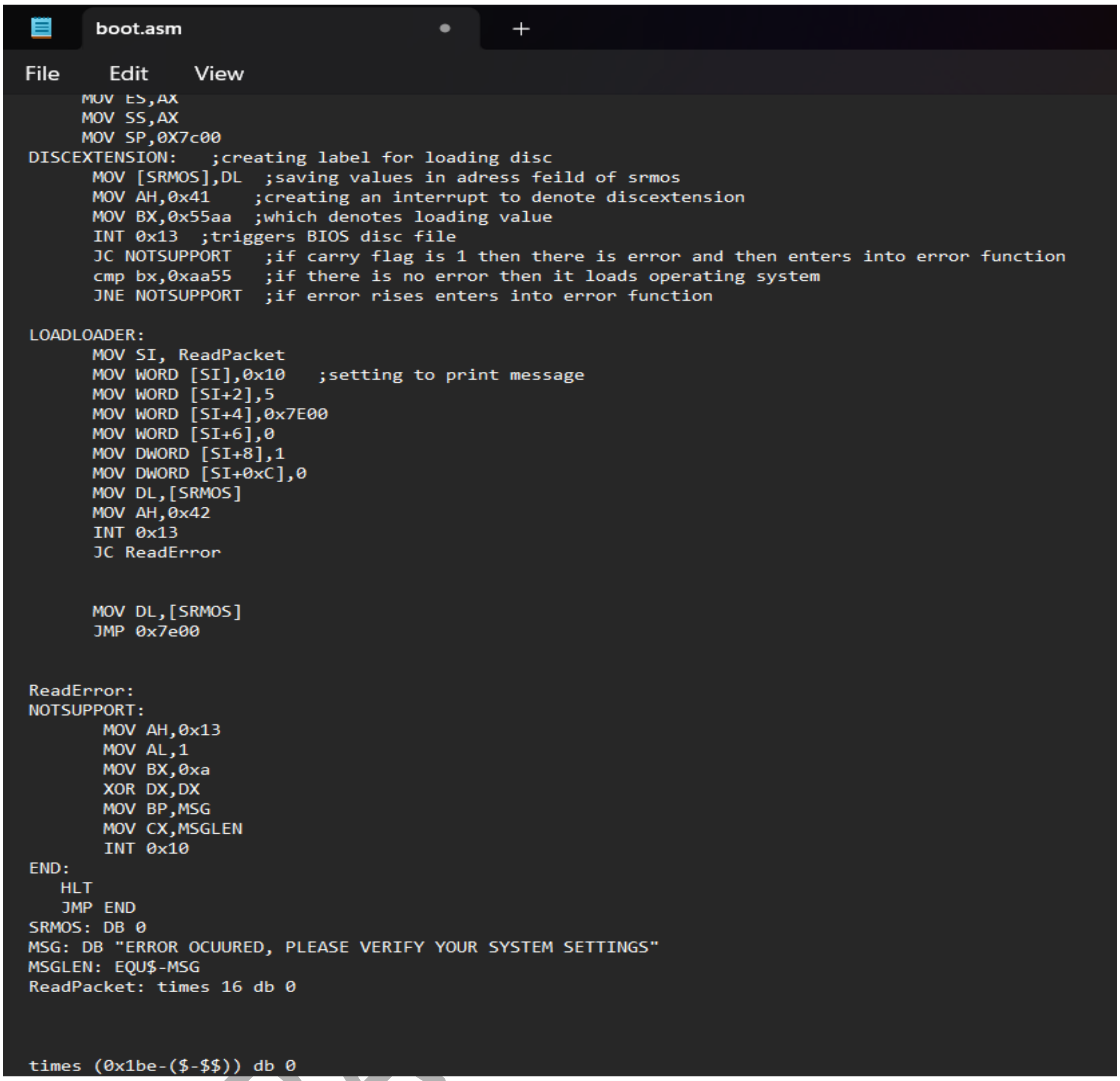
1. Static loading
2. Dynamic loading

In this project we are designing static loading.

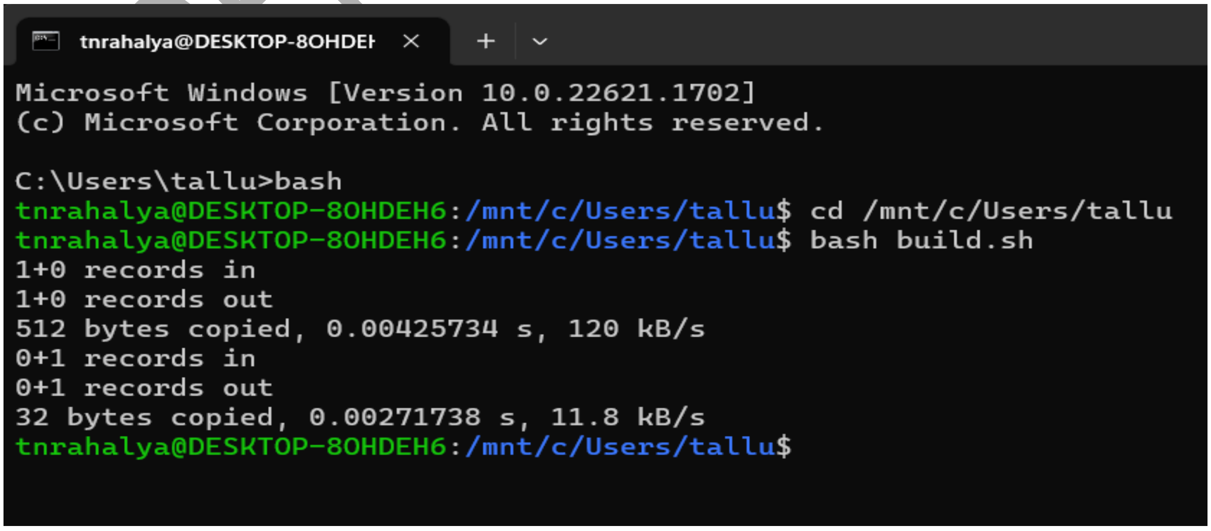
Program :



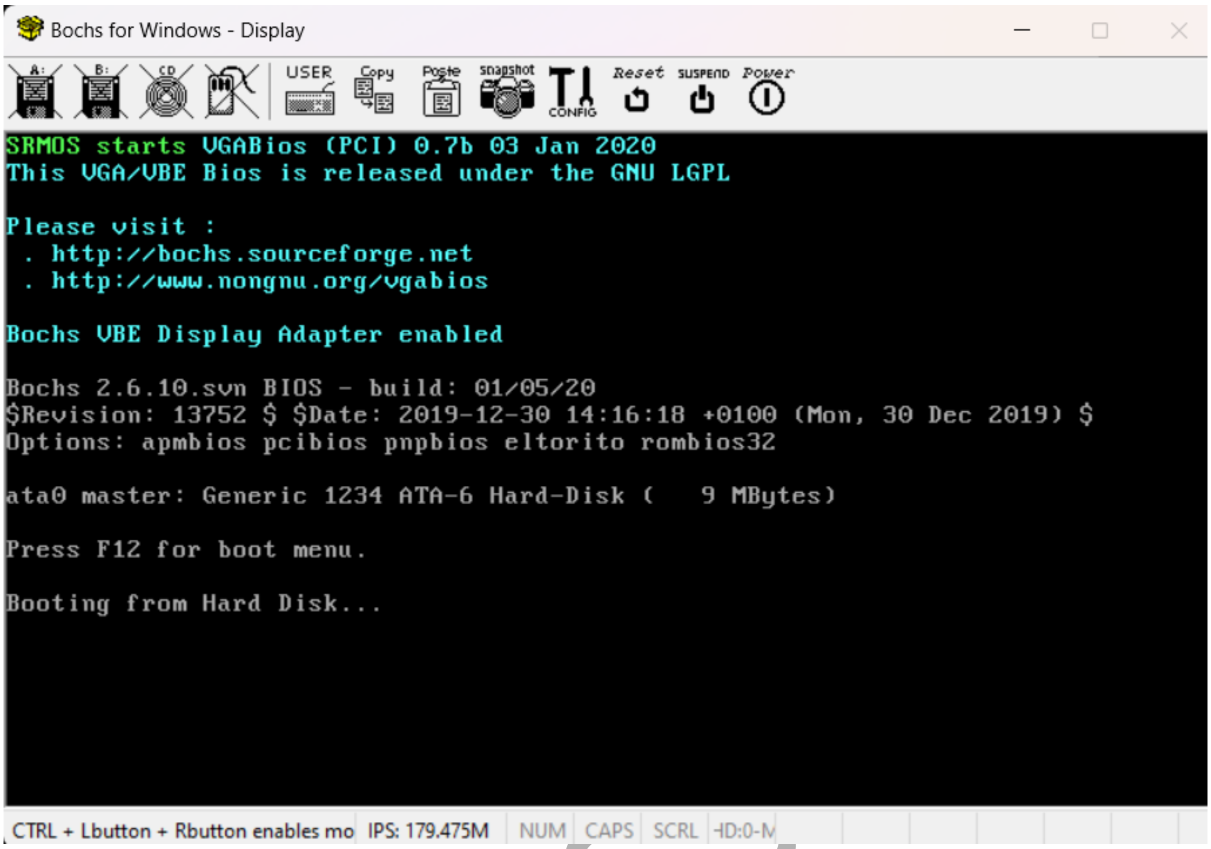
```
[BITS 16]
[ORG 0x7e00]

start:
      mov  ah,0x13
      mov  al,1
      mov  bx,0xa
      xor  dx,dx
      mov  bp,Message
      mov  cx,MessageLen
      int  0x10

End:
      hlt
      jmp  End

Message:     db "SRMOS starts"
MessageLen:  equ $-Message
```



```
      MOV ES,AX
      MOV SS,AX
      MOV SP,0X7c00
DISCEXTENSION:   ;creating label for loading disc
       MOV [SRMOS],DL  ;saving values in adress feild of srmos
       MOV AH,0x41    ;creating an interrupt to denote discextension
       MOV BX,0x55aa  ;which denotes loading value
       INT 0x13  ;triggers BIOS disc file
       JC NOTSUPPORT   ;if carry flag is 1 then there is error and then enters into error function
       cmp bx,0xaa55   ;if there is no error then it loads operating system
       JNE NOTSUPPORT  ;if error rises enters into error function

LOADLOADER:
       MOV SI, ReadPacket
       MOV WORD [SI],0x10   ;setting to print message
       MOV WORD [SI+2],5
       MOV WORD [SI+4],0x7E00
       MOV WORD [SI+6],0
       MOV DWORD [SI+8],1
       MOV DWORD [SI+0xC],0
       MOV DL,[SRMOS]
       MOV AH,0x42
       INT 0x13
       JC ReadError


       MOV DL,[SRMOS]
       JMP 0x7e00


ReadError:
NOTSUPPORT:
        MOV AH,0x13
        MOV AL,1
        MOV BX,0xa
        XOR DX,DX
        MOV BP,MSG
        MOV CX,MSGLEN
        INT 0x10
END:
   HLT
   JMP END
SRMOS: DB 0
MSG: DB "ERROR OCUURED, PLEASE VERIFY YOUR SYSTEM SETTINGS"
MSGLEN: EQU$-MSG
ReadPacket: times 16 db 0


times (0x1be-($-$$)) db 0
```



```
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\tallu>bash
tnrahalya@DESKTOP-8OHDEH6:/mnt/c/Users/tallu$ cd /mnt/c/Users/tallu
tnrahalya@DESKTOP-8OHDEH6:/mnt/c/Users/tallu$ bash build.sh
1+0 records in
1+0 records out
512 bytes copied, 0.00425734 s, 120 kB/s
0+1 records in
0+1 records out
32 bytes copied, 0.00271738 s, 11.8 kB/s
tnrahalya@DESKTOP-8OHDEH6:/mnt/c/Users/tallu$
```

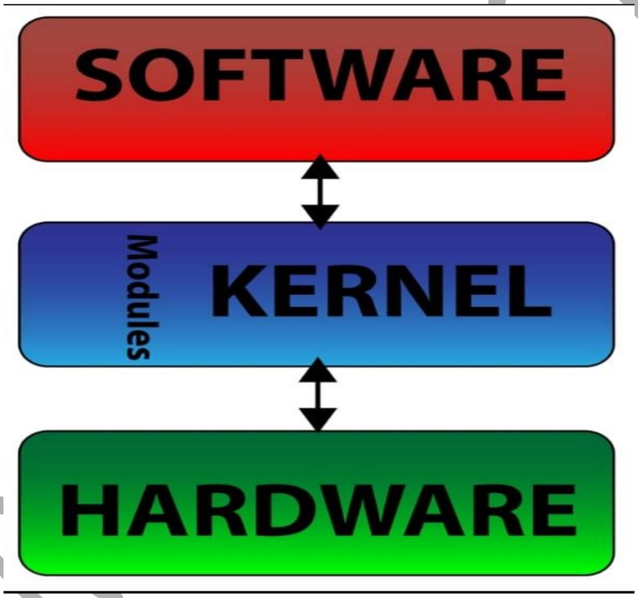


```
SRMOS starts VGABios (PCI) 0.7b 03 Jan 2020
This VGA/VBE Bios is released under the GNU LGPL

Please visit :
 . http://bochs.sourceforge.net
 . http://www.nongnu.org/vgabios

Bochs VBE Display Adapter enabled

Bochs 2.6.10.svn BIOS - build: 01/05/20
$Revision: 13752 $ $Date: 2019-12-30 14:16:18 +0100 (Mon, 30 Dec 2019) $
Options: apmbios pcibios pnpbios eltorito rombios32

ata0 master: Generic 1234 ATA-6 Hard-Disk (    9 MBytes)

Press F12 for boot menu.

Booting from Hard Disk...
```

KERNEL

The kernel is a program which runs at all time when operating system is on.

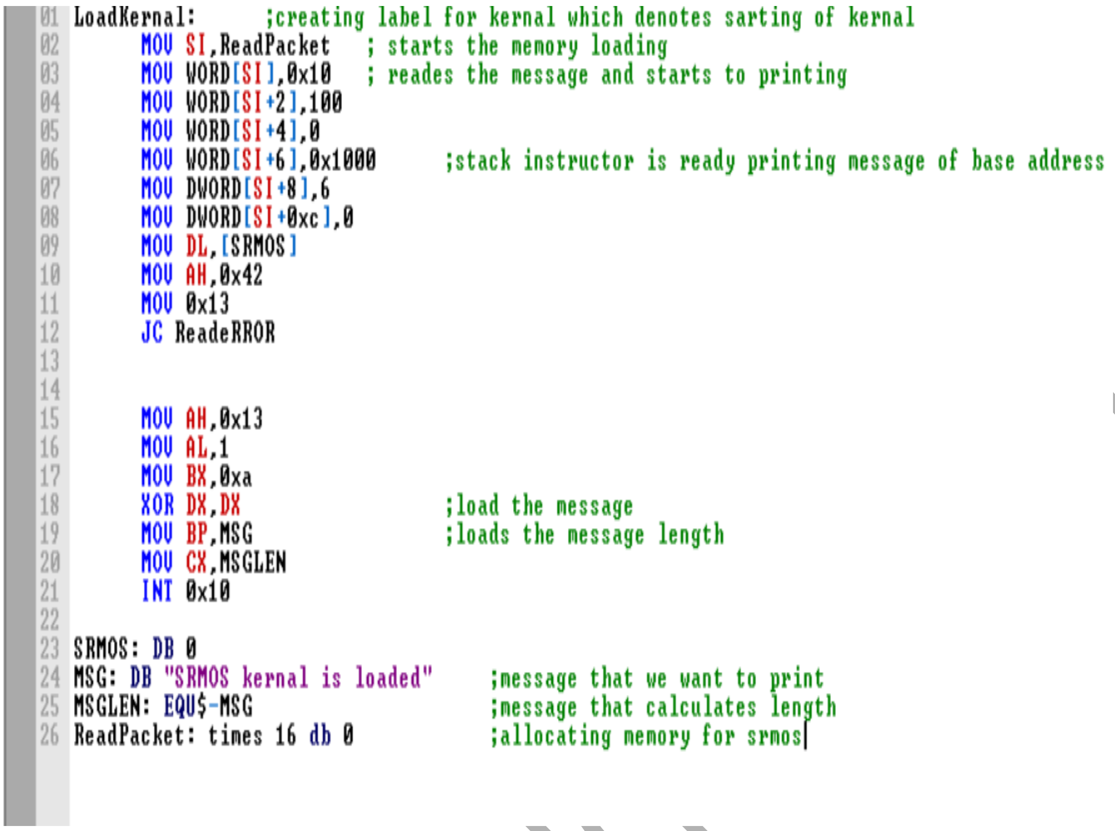
It is a core of operating system and it has complete control on everything in the system. It is portion of operating system code that is always resident in memory and facilities interactions between hardware and software components.



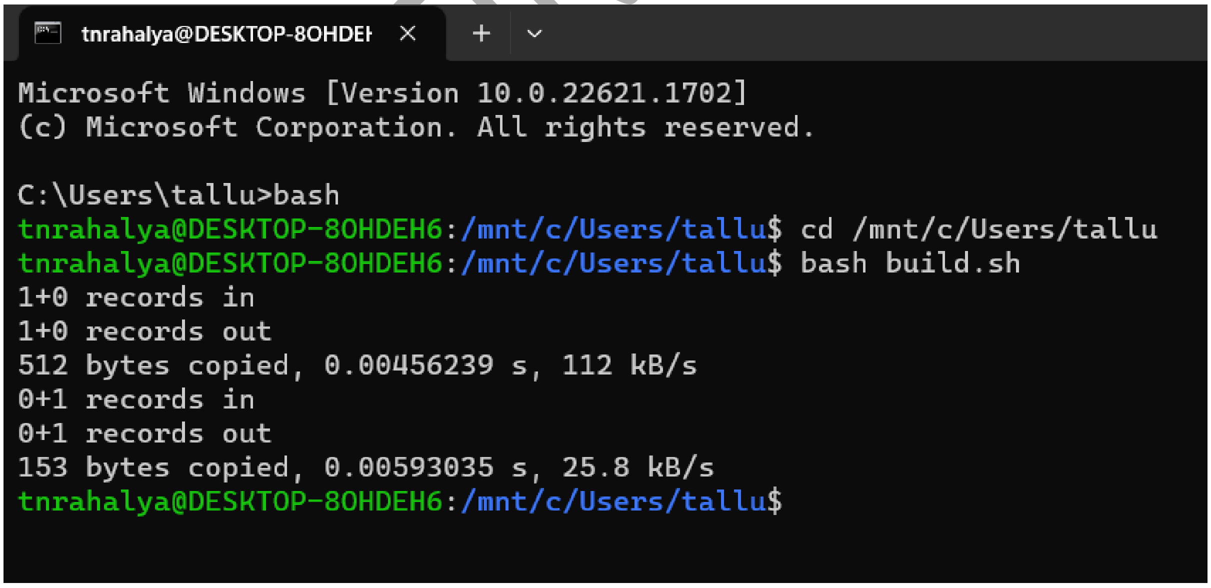
In this project our kernel which holds address of loader file(loading disc) and booting file. The kernel main purpose is to acts intermediate between operating system and hardware.

- It controls disc management
- Control memory management and task management
- Improve communication between user level application and hardware

Program for kernel :

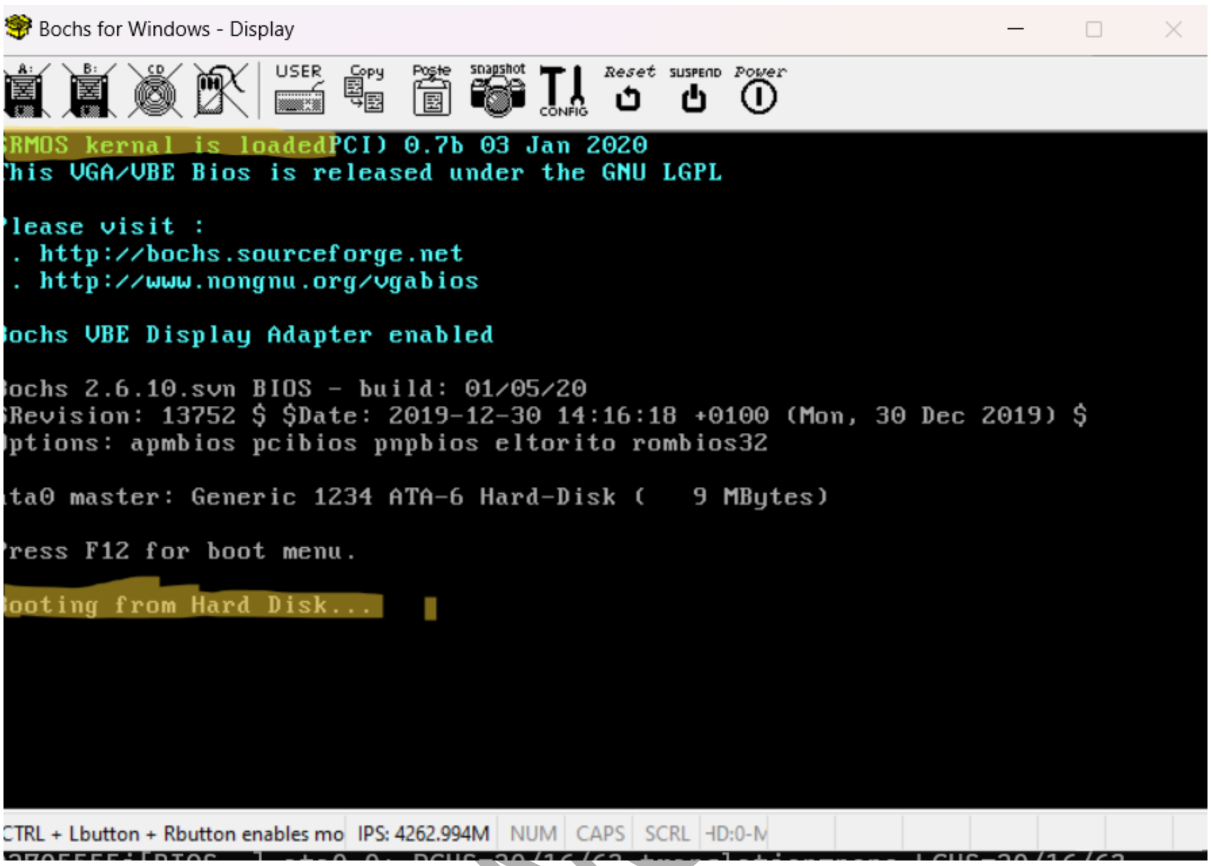


```
LoadKernal:      ;creating label for kernal which denotes sarting of kernal
      MOV SI,ReadPacket   ; starts the memory loading
      MOV WORD[SI],0x10   ; reades the message and starts to printing
      MOV WORD[SI+2],100
      MOV WORD[SI+4],0
      MOV WORD[SI+6],0x1000      ;stack instructor is ready printing message of base address
      MOV DWORD[SI+8],6
      MOV DWORD[SI+0xc],0
      MOV DL,[SRMOS]
      MOV AH,0x42
      MOV 0x13
      JC ReadeRROR


      MOV AH,0x13
      MOV AL,1
      MOV BX,0xa
      XOR DX,DX                  ;load the message
      MOV BP,MSG                 ;loads the message length
      MOV CX,MSGLEN
      INT 0x10

SRMOS: DB 0
MSG: DB "SRMOS kernal is loaded"      ;message that we want to print
MSGLEN: EQU$-MSG                      ;message that calculates length
ReadPacket: times 16 db 0             ;allocating memory for srmos
```



```
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\tallu>bash
tnrahalya@DESKTOP-8OHDEH6:/mnt/c/Users/tallu$ cd /mnt/c/Users/tallu
tnrahalya@DESKTOP-8OHDEH6:/mnt/c/Users/tallu$ bash build.sh
1+0 records in
1+0 records out
512 bytes copied, 0.00456239 s, 112 kB/s
0+1 records in
0+1 records out
153 bytes copied, 0.00593035 s, 25.8 kB/s
tnrahalya@DESKTOP-8OHDEH6:/mnt/c/Users/tallu$
```

My kernel :



```
RMOS kernal is loadedPCI) 0.7b 03 Jan 2020
his VGA/VBE Bios is released under the GNU LGPL

lease visit :
. http://bochs.sourceforge.net
. http://www.nongnu.org/vgabios

ochs VBE Display Adapter enabled

ochs 2.6.10.svn BIOS - build: 01/05/20
Revision: 13752 $ $Date: 2019-12-30 14:16:18 +0100 (Mon, 30 Dec 2019) $
ptions: apmbios pcibios pnpbios eltorito rombios32

ta0 master: Generic 1234 ATA-6 Hard-Disk (   9 MBytes)

ress F12 for boot menu.

ooting from Hard Disk...
```

INTERRUPT HANDLING

Interrupts :

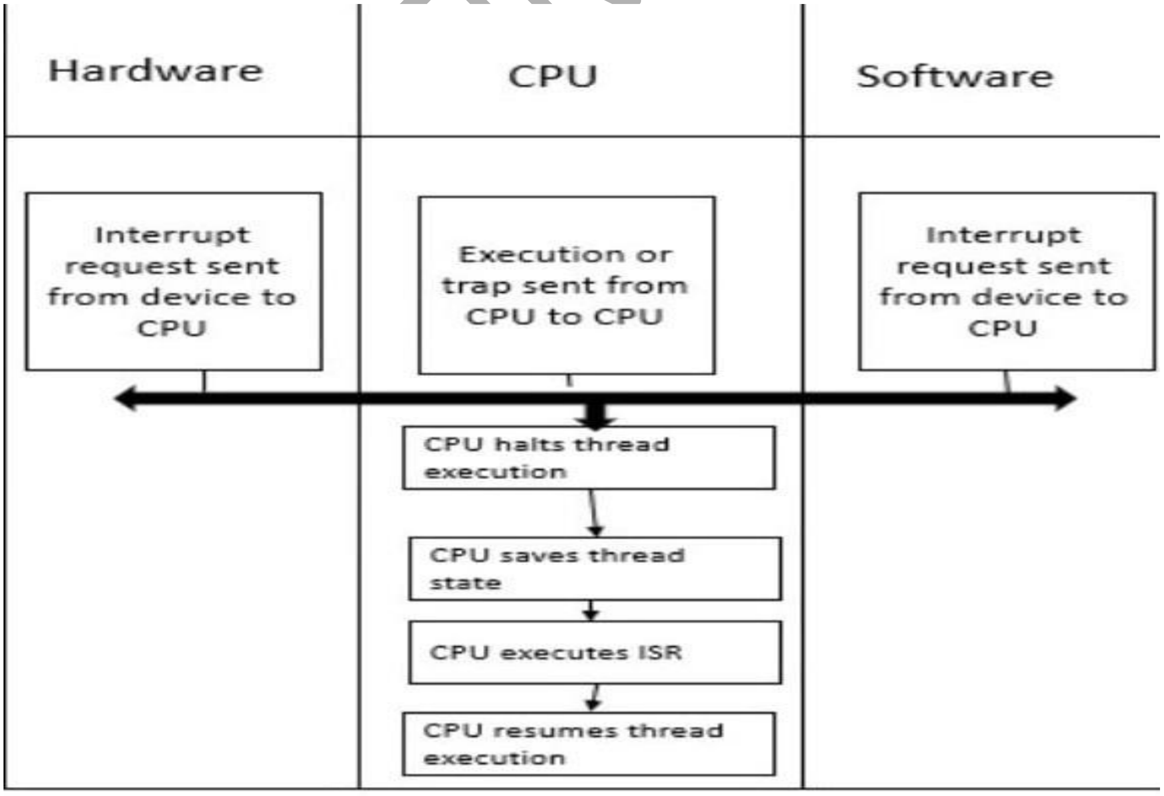
Interrupts are signals which are generated by the software or hardware to stop or execute the process or program.

Interrupt Handling :

The way of executing interrupts is known as interrupt handling.

“Interrupt service routine (ISR)” or Interrupt handler are responsible for the interrupts.

“ IRET , RET “ are responsible for handling interrupts.


Hardware
CPU
Software
Interrupt request sent from device to CPU
Execution or trap sent from CPU to CPU
Interrupt request sent from device to CPU
CPU halts thread execution
CPU saves thread state
CPU executes ISR
CPU resumes thread execution

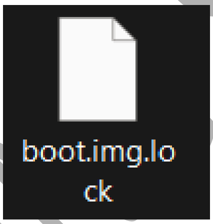
OUTPUT

FEATURES OF MY OPERATING SYSTEM :

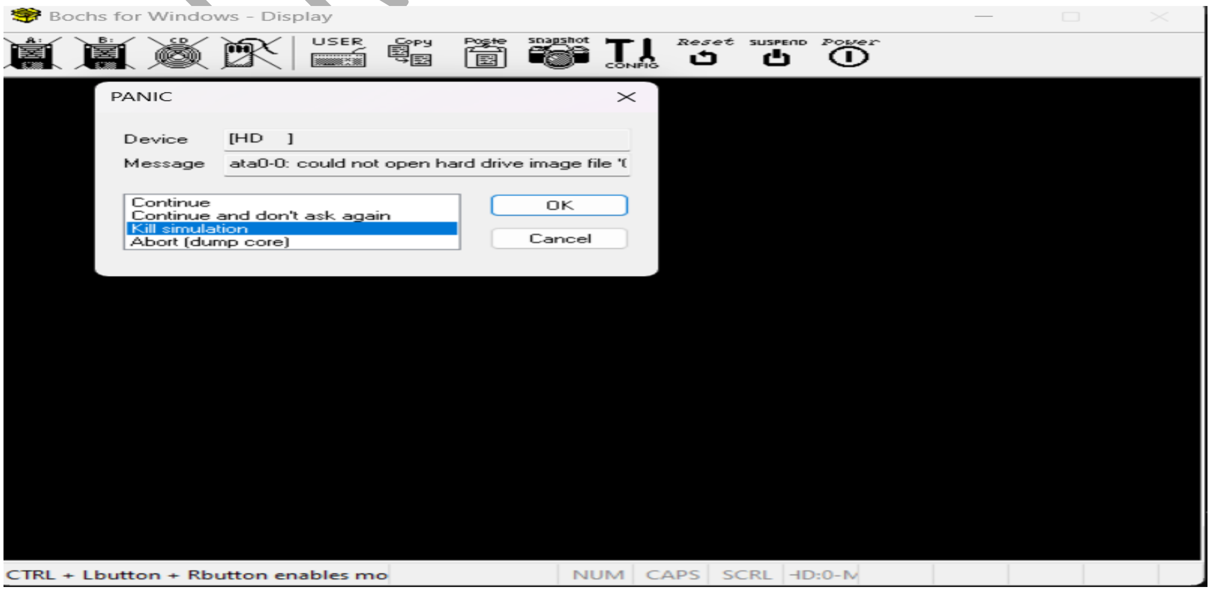
LOCKING :

Our operating system will be locked automatically by creating a file ("boot.img.lock" which is responsible for locking file. It will destroy previous memory and does not allocate memory for hardware to stop the operating system.

FILE :



This file is responsible for locking.



PANIC

Device [HD]

Message ata0-0: could not open hard drive image file '(

Continue
Continue and don't ask again
Kill simulation
Abort (dump core)

OK
Cancel

CTRL + Lbutton + Rbutton enables mo | NUM | CAPS | SCRL | HD:0-M

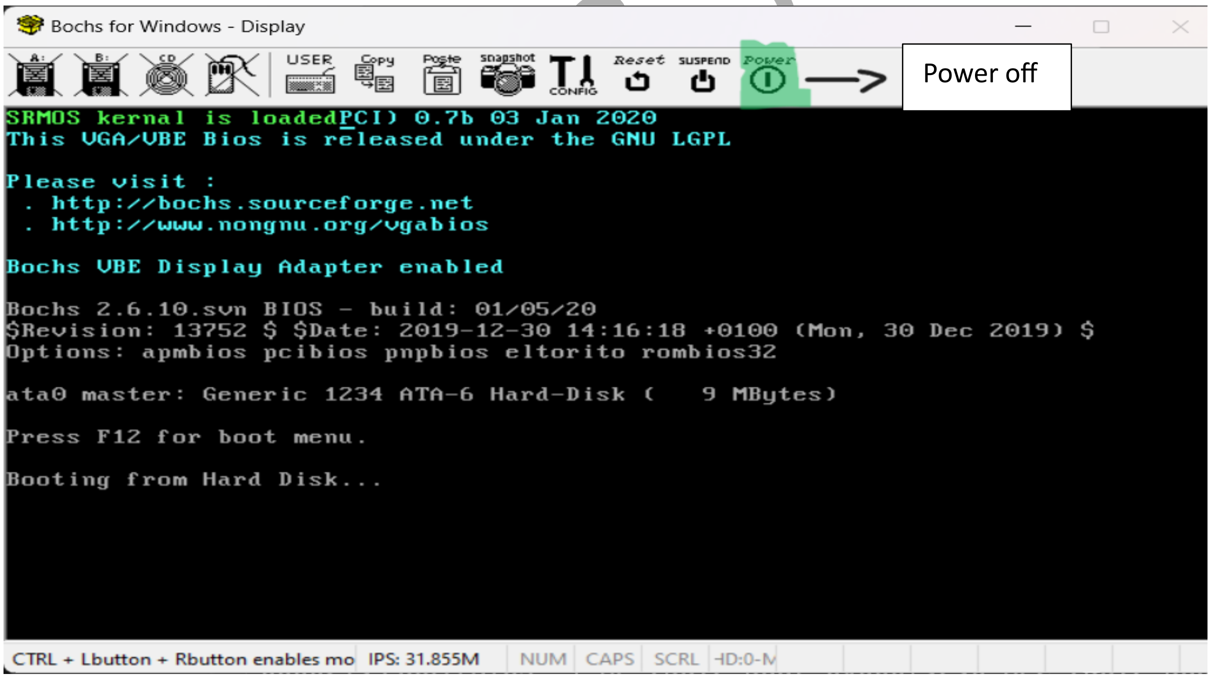
The above screen will and simply signify that operating system is locked and the memory is does not allocated.

" ata0-0: could not open hard drive image file 'C:\Users\tallu\boot.img' "

The above message will print.

POWER OFF :

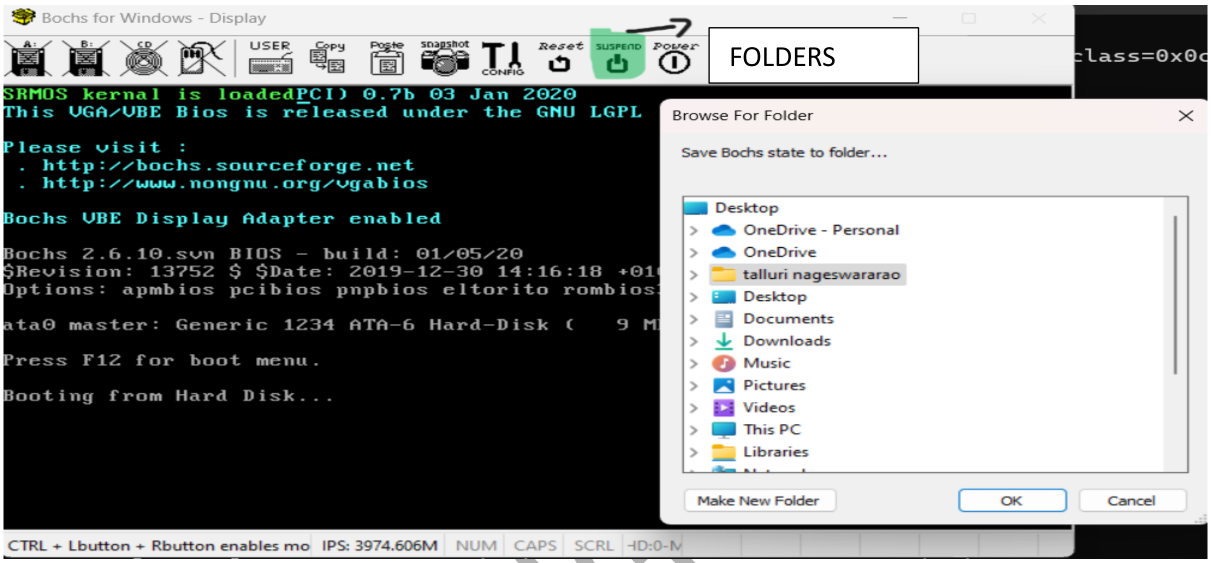
The "power off " option will be displayed which will turn off our operating system after few seconds. If want to start our operating system again then we need load all the files again, Then our operating will turn on .



```
SRMOS kernal is loadedPCI) 0.7b 03 Jan 2020
This VGA/VBE Bios is released under the GNU LGPL

Please visit :
 . http://bochs.sourceforge.net
 . http://www.nongnu.org/vgabios

Bochs VBE Display Adapter enabled

Bochs 2.6.10.svn BIOS - build: 01/05/20
$Revision: 13752 $ $Date: 2019-12-30 14:16:18 +0100 (Mon, 30 Dec 2019) $
Options: apmbios pcibios pnpbios eltorito rombios32

ata0 master: Generic 1234 ATA-6 Hard-Disk (   9 MBytes)

Press F12 for boot menu.

Booting from Hard Disk...
```

SAVING AND BROWSING FOLDERS AND FILES :

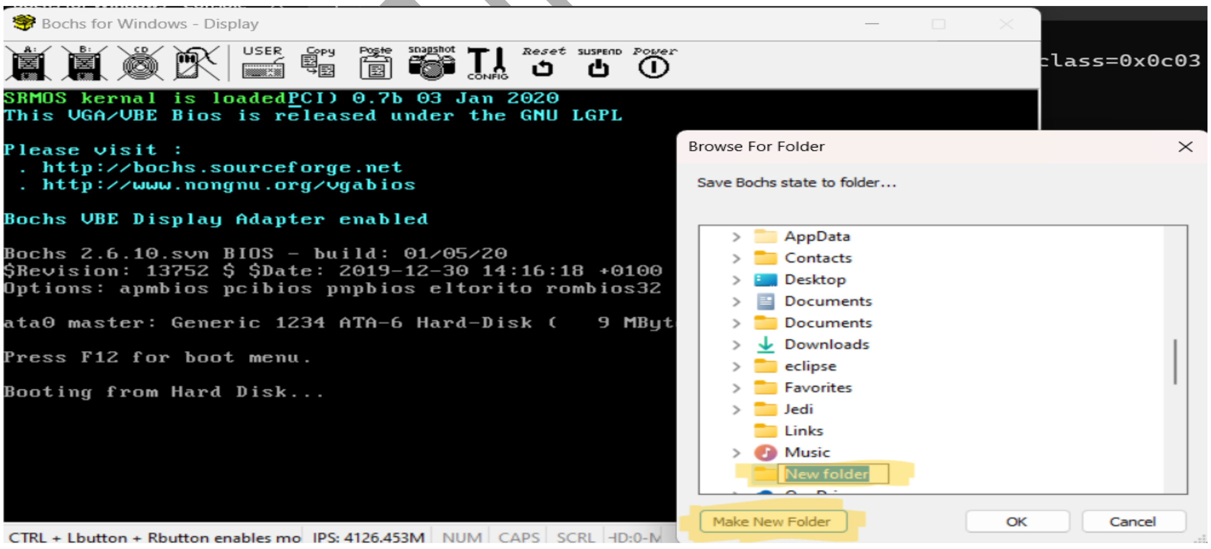
In this project we are allowing to create own file or folders and browsing files and folders. In this project we are providing option “suspend” to operate these files.



```
SRMOS kernal is loadedPCI) 0.7b 03 Jan 2020
This VGA/VBE Bios is released under the GNU LGPL

Please visit :
 . http://bochs.sourceforge.net
 . http://www.nongnu.org/vgabios

Bochs VBE Display Adapter enabled

Bochs 2.6.10.svn BIOS - build: 01/05/20
$Revision: 13752 $ $Date: 2019-12-30 14:16:18 +01
Options: apmbios pcibios pnpbios eltorito rombios

ata0 master: Generic 1234 ATA-6 Hard-Disk (   9 M

Press F12 for boot menu.

Booting from Hard Disk...
```

```
CTRL + Lbutton + Rbutton enables mo  IPS: 3974.606M  NUM  CAPS  SCRL  HD:0-M
```

MAKING NEW FOLDER :

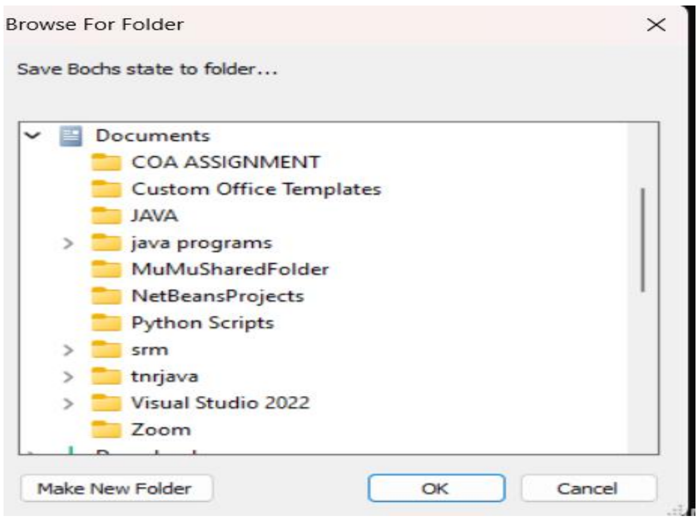


```
SRMOS kernal is loadedPCI) 0.7b 03 Jan 2020
This VGA/VBE Bios is released under the GNU LGPL

Please visit :
 . http://bochs.sourceforge.net
 . http://www.nongnu.org/vgabios

Bochs VBE Display Adapter enabled

Bochs 2.6.10.svn BIOS - build: 01/05/20
$Revision: 13752 $ $Date: 2019-12-30 14:16:18 +0100
Options: apmbios pcibios pnpbios eltorito rombios32

ata0 master: Generic 1234 ATA-6 Hard-Disk (   9 MByt

Press F12 for boot menu.

Booting from Hard Disk...
```

```
CTRL + Lbutton + Rbutton enables mo  IPS: 4126.453M  NUM  CAPS  SCRL  HD:0-M
```

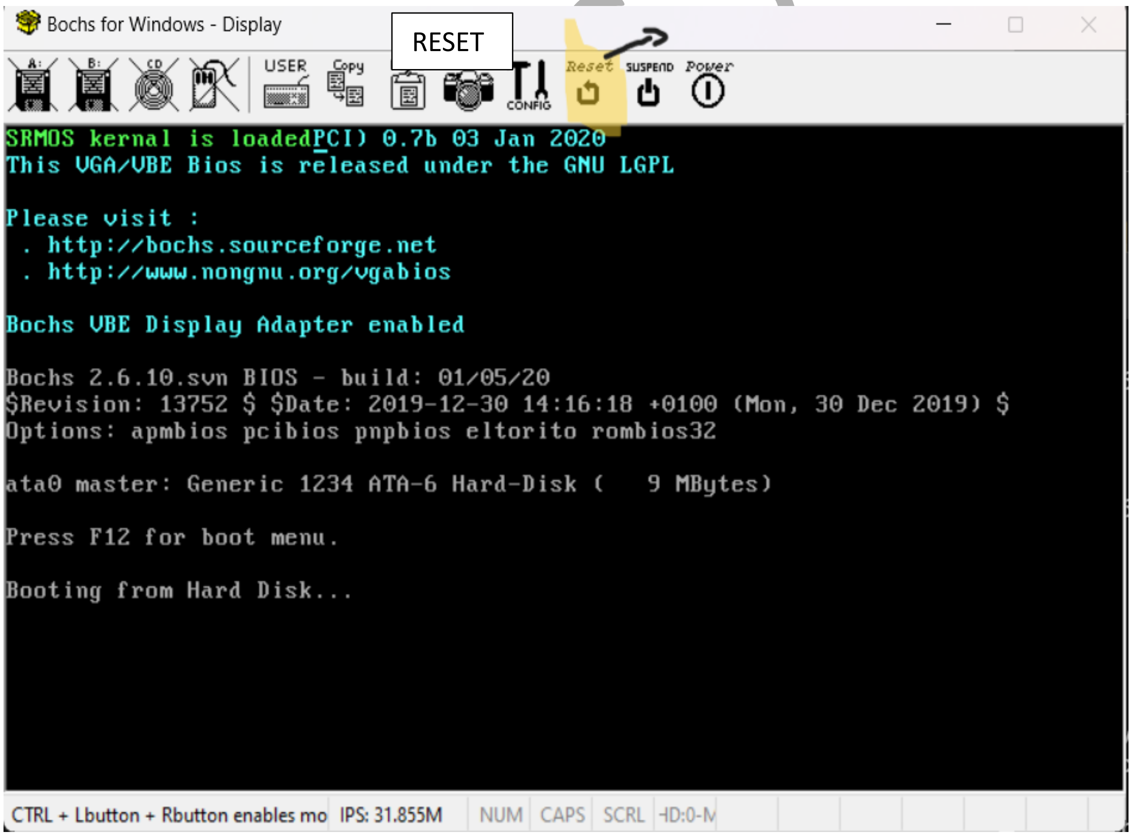
The above coloured thing is to create folders and also provide all option to manipulate files.

NAVIGATING TO FILES :



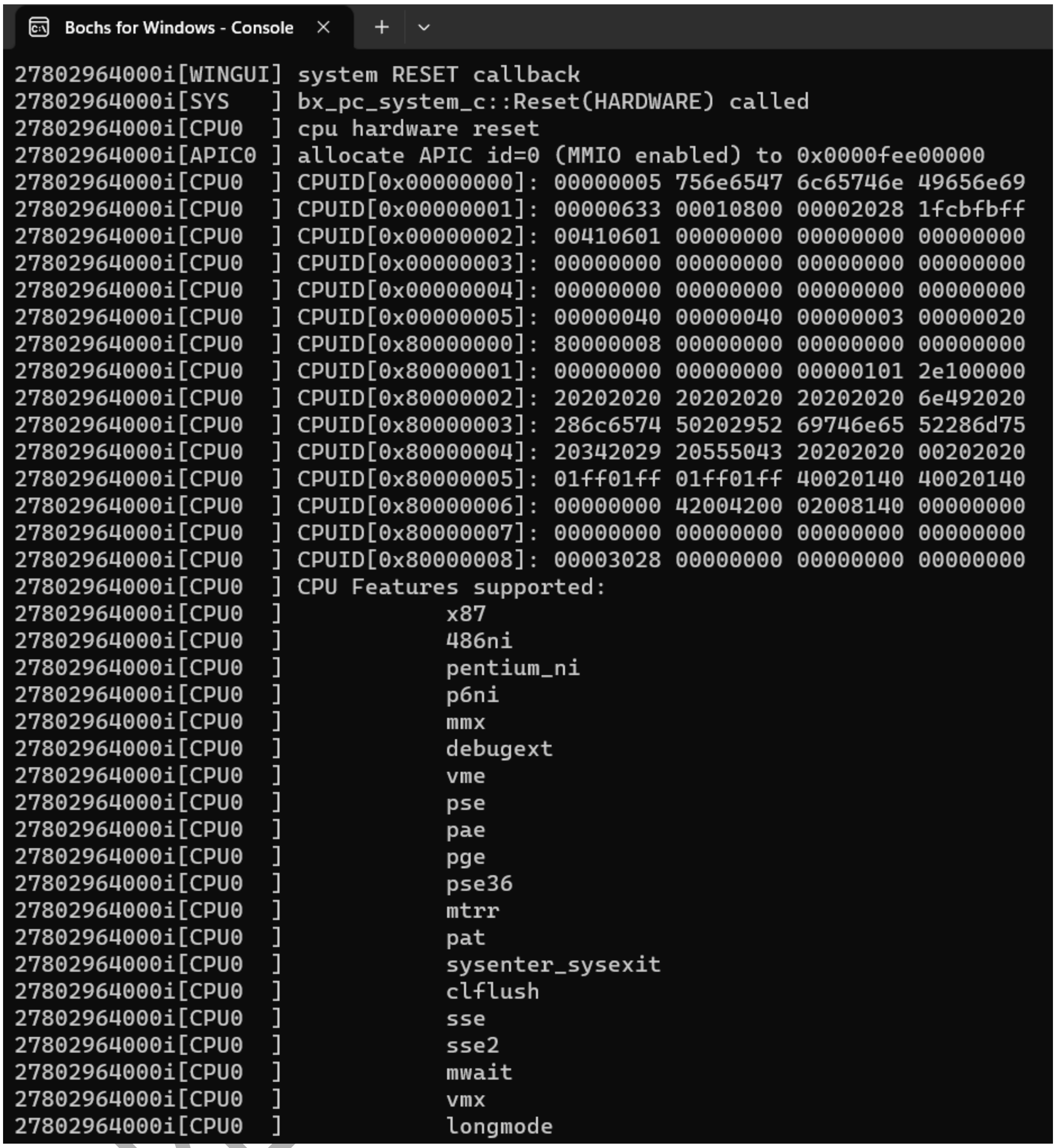
RESET :

The operating system will reset



```
SRMOS kernal is loadedPCI) 0.7b 03 Jan 2020
This VGA/VBE Bios is released under the GNU LGPL

Please visit :
 . http://bochs.sourceforge.net
 . http://www.nongnu.org/vgabios

Bochs VBE Display Adapter enabled

Bochs 2.6.10.svn BIOS - build: 01/05/20
$Revision: 13752 $ $Date: 2019-12-30 14:16:18 +0100 (Mon, 30 Dec 2019) $
Options: apmbios pcibios pnpbios eltorito rombios32

ata0 master: Generic 1234 ATA-6 Hard-Disk (   9 MBytes)

Press F12 for boot menu.

Booting from Hard Disk...
```

CTRL + Lbutton + Rbutton enables mo IPS: 31.855M NUM CAPS SCRL HD:0-M

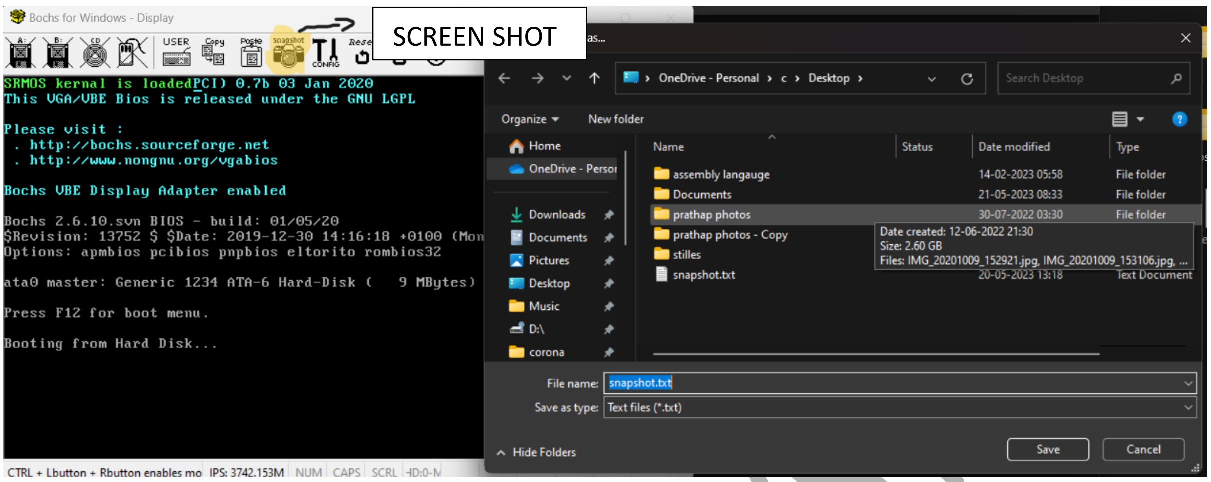


```
27802964000i[WINGUI] system RESET callback
27802964000i[SYS   ] bx_pc_system_c::Reset(HARDWARE) called
27802964000i[CPU0  ] cpu hardware reset
27802964000i[APIC0 ] allocate APIC id=0 (MMIO enabled) to 0x0000fee00000
27802964000i[CPU0  ] CPUID[0x00000000]: 00000005 756e6547 6c65746e 49656e69
27802964000i[CPU0  ] CPUID[0x00000001]: 00000633 00010800 00002028 1fcbfbff
27802964000i[CPU0  ] CPUID[0x00000002]: 00410601 00000000 00000000 00000000
27802964000i[CPU0  ] CPUID[0x00000003]: 00000000 00000000 00000000 00000000
27802964000i[CPU0  ] CPUID[0x00000004]: 00000000 00000000 00000000 00000000
27802964000i[CPU0  ] CPUID[0x00000005]: 00000040 00000040 00000003 00000020
27802964000i[CPU0  ] CPUID[0x80000000]: 80000008 00000000 00000000 00000000
27802964000i[CPU0  ] CPUID[0x80000001]: 00000000 00000000 00000101 2e100000
27802964000i[CPU0  ] CPUID[0x80000002]: 20202020 20202020 20202020 6e492020
27802964000i[CPU0  ] CPUID[0x80000003]: 286c6574 50202952 69746e65 52286d75
27802964000i[CPU0  ] CPUID[0x80000004]: 20342029 20555043 20202020 00202020
27802964000i[CPU0  ] CPUID[0x80000005]: 01ff01ff 01ff01ff 40020140 40020140
27802964000i[CPU0  ] CPUID[0x80000006]: 00000000 42004200 02008140 00000000
27802964000i[CPU0  ] CPUID[0x80000007]: 00000000 00000000 00000000 00000000
27802964000i[CPU0  ] CPUID[0x80000008]: 00003028 00000000 00000000 00000000
27802964000i[CPU0  ] CPU Features supported:
27802964000i[CPU0  ]            x87
27802964000i[CPU0  ]            486ni
27802964000i[CPU0  ]            pentium_ni
27802964000i[CPU0  ]            p6ni
27802964000i[CPU0  ]            mmx
27802964000i[CPU0  ]            debugext
27802964000i[CPU0  ]            vme
27802964000i[CPU0  ]            pse
27802964000i[CPU0  ]            pae
27802964000i[CPU0  ]            pge
27802964000i[CPU0  ]            pse36
27802964000i[CPU0  ]            mtrr
27802964000i[CPU0  ]            pat
27802964000i[CPU0  ]            sysenter_sysexit
27802964000i[CPU0  ]            clflush
27802964000i[CPU0  ]            sse
27802964000i[CPU0  ]            sse2
27802964000i[CPU0  ]            mwait
27802964000i[CPU0  ]            vmx
27802964000i[CPU0  ]            longmode
```

The memory commands for resetting operating system with all the files and folders

SNAPCHAT OR SCREENSHOT :

In our project for camera option it will provide snapchat option. It will be save in folders as text document are what ever we want format.



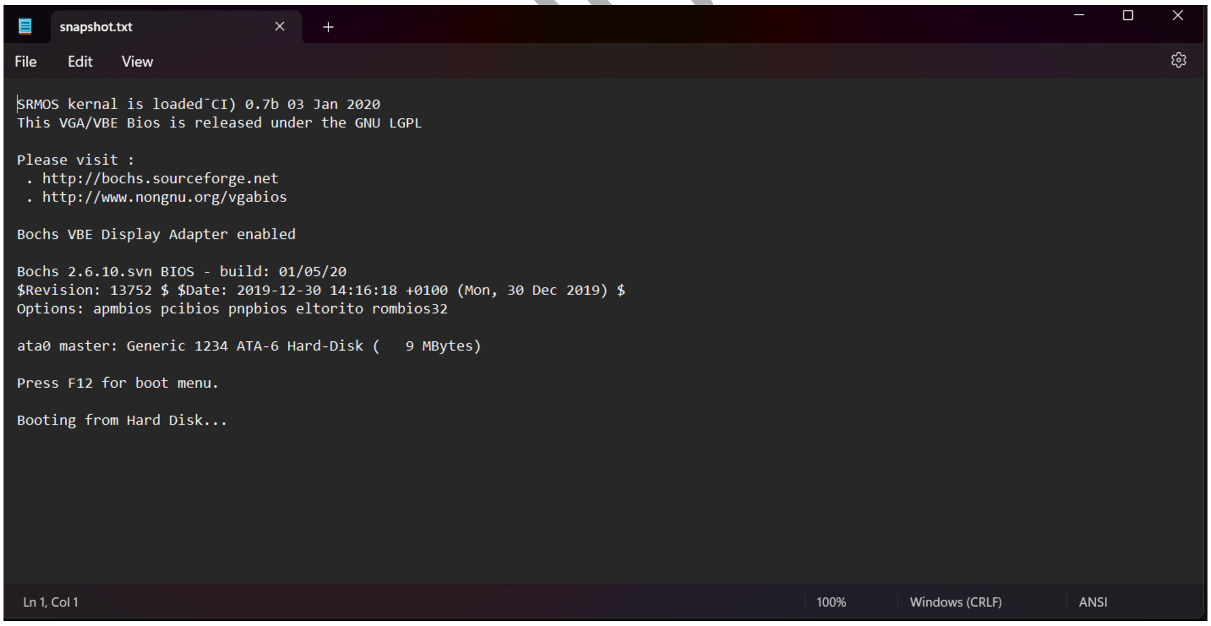
```
SRMOS kernal is loadedPCI) 0.7b 03 Jan 2020
This VGA/VBE Bios is released under the GNU LGPL

Please visit :
 . http://bochs.sourceforge.net
 . http://www.nongnu.org/vgabios

Bochs VBE Display Adapter enabled

Bochs 2.6.10.svn BIOS - build: 01/05/20
$Revision: 13752 $ $Date: 2019-12-30 14:16:18 +0100 (Mon
Options: apmbios pcibios pnpbios eltorito rombios32

ata0 master: Generic 1234 ATA-6 Hard-Disk (   9 MBytes)

Press F12 for boot menu.

Booting from Hard Disk...
```

Name	Status	Date modified	Type
assembly langauge		14-02-2023 05:58	File folder
Documents		21-05-2023 08:33	File folder
prathap photos		30-07-2022 03:30	File folder
prathap photos - Copy			
stilles			
snapshot.txt		20-05-2023 13:18	Text Document

Date created: 12-06-2022 21:30
Size: 2.60 GB
Files: IMG_20201009_152921.jpg, IMG_20201009_153106.jpg, ...

File name: snapshot.txt
Save as type: Text files (*.txt)

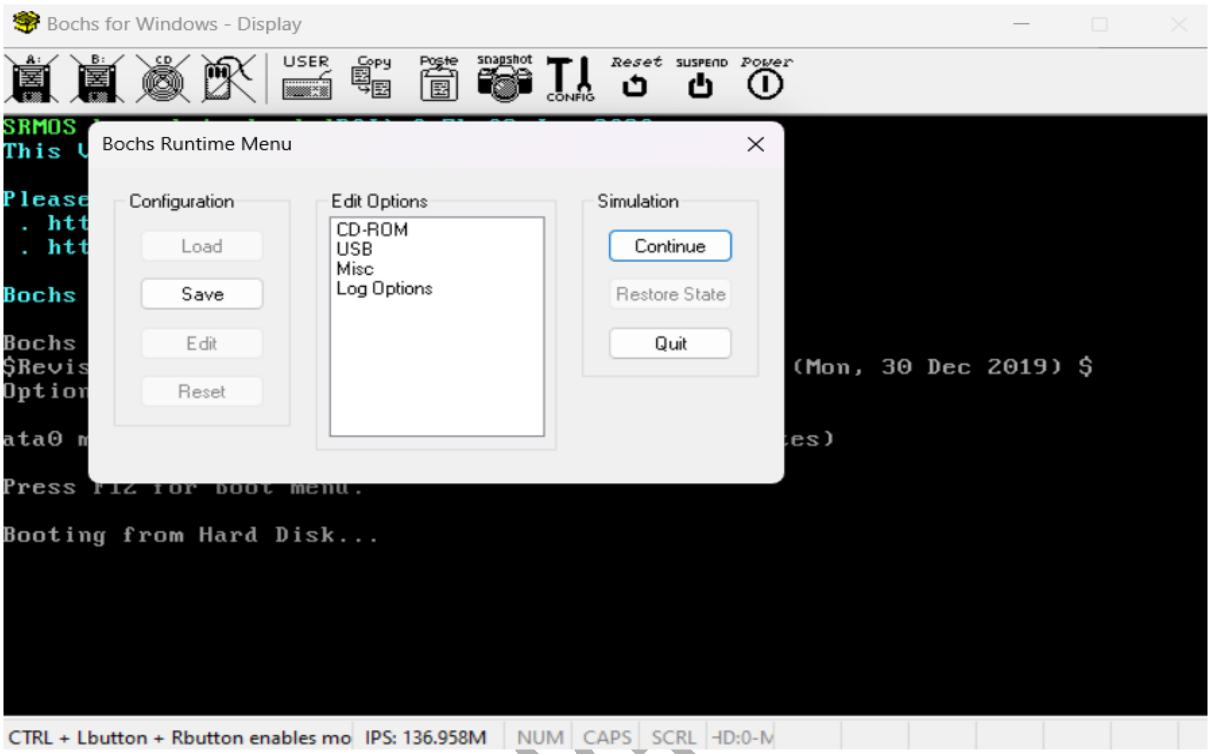
OUR FIRST SCREEN SHOT :



```
SRMOS kernal is loaded¯CI) 0.7b 03 Jan 2020
This VGA/VBE Bios is released under the GNU LGPL

Please visit :
 . http://bochs.sourceforge.net
 . http://www.nongnu.org/vgabios

Bochs VBE Display Adapter enabled

Bochs 2.6.10.svn BIOS - build: 01/05/20
$Revision: 13752 $ $Date: 2019-12-30 14:16:18 +0100 (Mon, 30 Dec 2019) $
Options: apmbios pcibios pnpbios eltorito rombios32

ata0 master: Generic 1234 ATA-6 Hard-Disk (   9 MBytes)

Press F12 for boot menu.

Booting from Hard Disk...
```

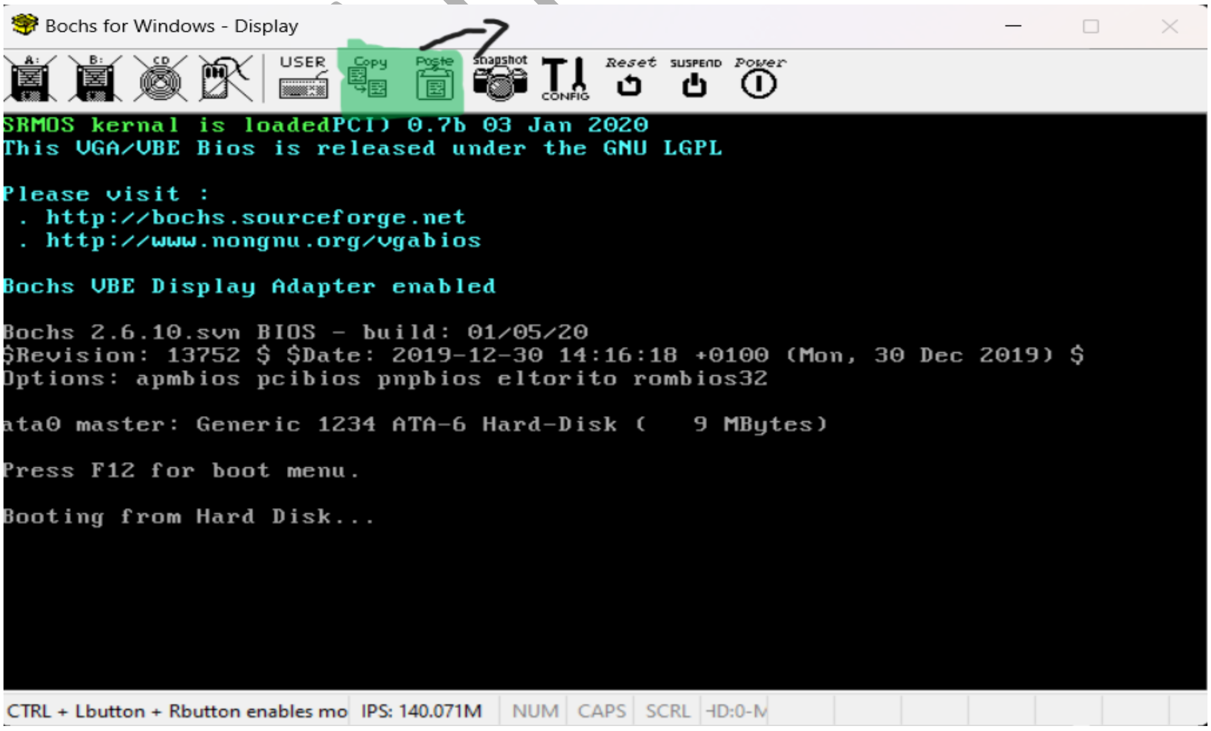
SETTINGS :

To change the settings we providing option “config” .



```
SRMOS
This U

Please
 . htt
 . htt

Bochs

Bochs
$Revis                                                      (Mon, 30 Dec 2019) $
Option

ata0 m                                                      es)

Press F12 for boot menu.

Booting from Hard Disk...
```

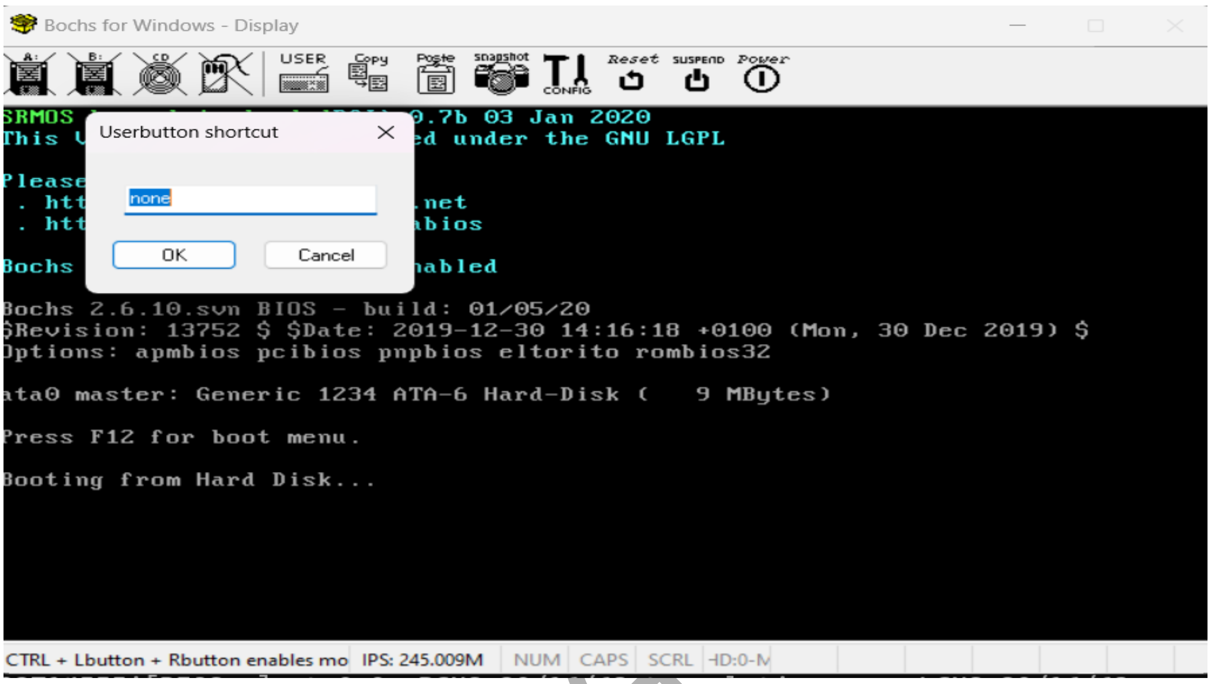
COPY AND PASTE :



```
SRMOS kernal is loadedPCI) 0.7b 03 Jan 2020
This VGA/VBE Bios is released under the GNU LGPL

Please visit :
 . http://bochs.sourceforge.net
 . http://www.nongnu.org/vgabios

Bochs VBE Display Adapter enabled

Bochs 2.6.10.svn BIOS - build: 01/05/20
$Revision: 13752 $ $Date: 2019-12-30 14:16:18 +0100 (Mon, 30 Dec 2019) $
Options: apmbios pcibios pnpbios eltorito rombios32

ata0 master: Generic 1234 ATA-6 Hard-Disk (   9 MBytes)

Press F12 for boot menu.

Booting from Hard Disk...
```

KEYBOARD :

In this project we are using wire keyboard to operate the operating system and wireless operating system is not worked.



```
SRMOS                    0.7b 03 Jan 2020
This U                   ed under the GNU LGPL

Please
 . htt                   .net
 . htt                   abios

Bochs                    nabled

Bochs 2.6.10.svn BIOS - build: 01/05/20
$Revision: 13752 $ $Date: 2019-12-30 14:16:18 +0100 (Mon, 30 Dec 2019) $
Options: apmbios pcibios pnpbios eltorito rombios32

ata0 master: Generic 1234 ATA-6 Hard-Disk (    9 MBytes)

Press F12 for boot menu.

Booting from Hard Disk...
```

The above mentioned things are features of our operating system.

REPORT :

In this project we are developing an operating system which is named as " SRMOS" and in this project we are providing some features. In this project we are providing basic information regarding to the creation of simple operating system. We are using windows and linux to develop another operating system using bochs software.

The features provided in this operating system are :

1. Locking and power off of the operating system
2. Creating and browsing files
3. Snapshot and screenshot
4. Settings
5. Copy and paste
6. Keyboard

By this project we an overview regarding operating system how it works , importance of operating system , advantages and disadvantages of operating system and new softwares and programming languages.

That's all about my project ....

THE END