

# **OPERATING SYSTEMS**

## **Lab-3: BOOT LOADER CURSOR, ALPHABET AND DISPLAY MESSAGE**

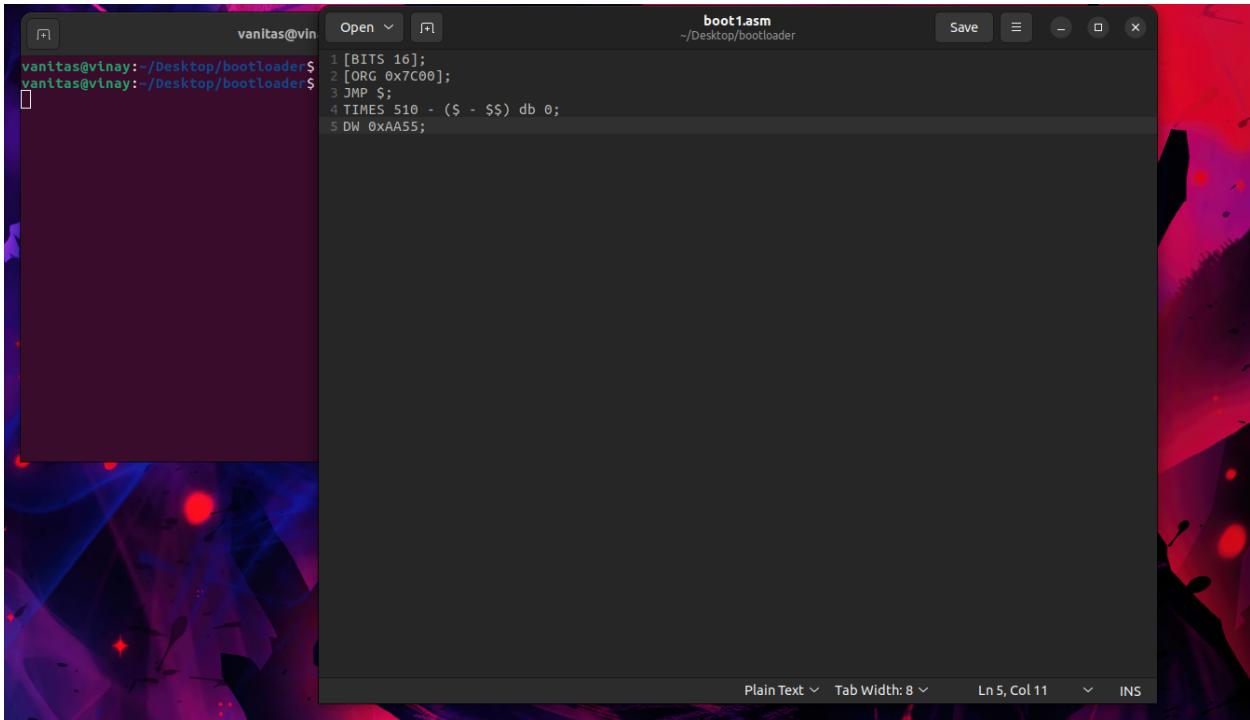
Name – Vinay Santosh Menon  
Registration Number - 20BAI1103

### **TASK:**

Creating a bootloader program in assembly language, compiling it and running it in a virtual machine.

## Empty bootloader cursor:

Write the below assembly code in a text editor and save it as a .asm file:

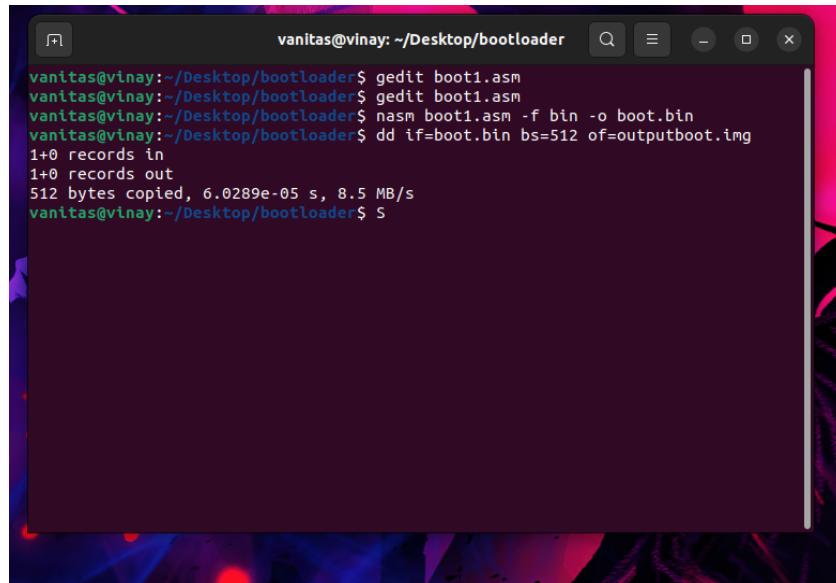


The screenshot shows a terminal window titled "boot1.asm" with the path "/Desktop/bootloader". The code in the editor is:

```
1 [BITS 16];
2 [ORG 0x7C00];
3 JMP $;
4 TIMES 510 - ($ - $$) db 0;
5 DW 0xAA55;
```

The terminal window also displays the command prompt "vanitas@vinay: ~/Desktop/bootloader\$".

Compile the code using nasm in the Linux terminal to convert it first into a binary file and then into an iso img file. The commands below are used to achieve this.

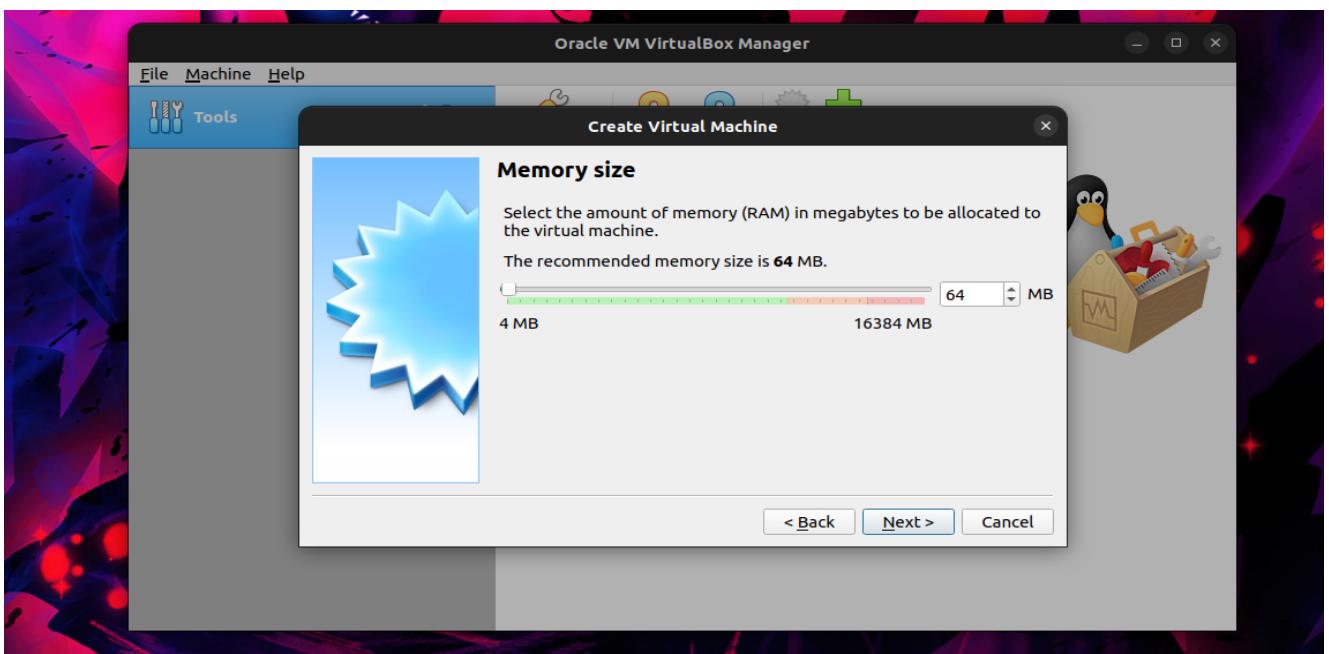
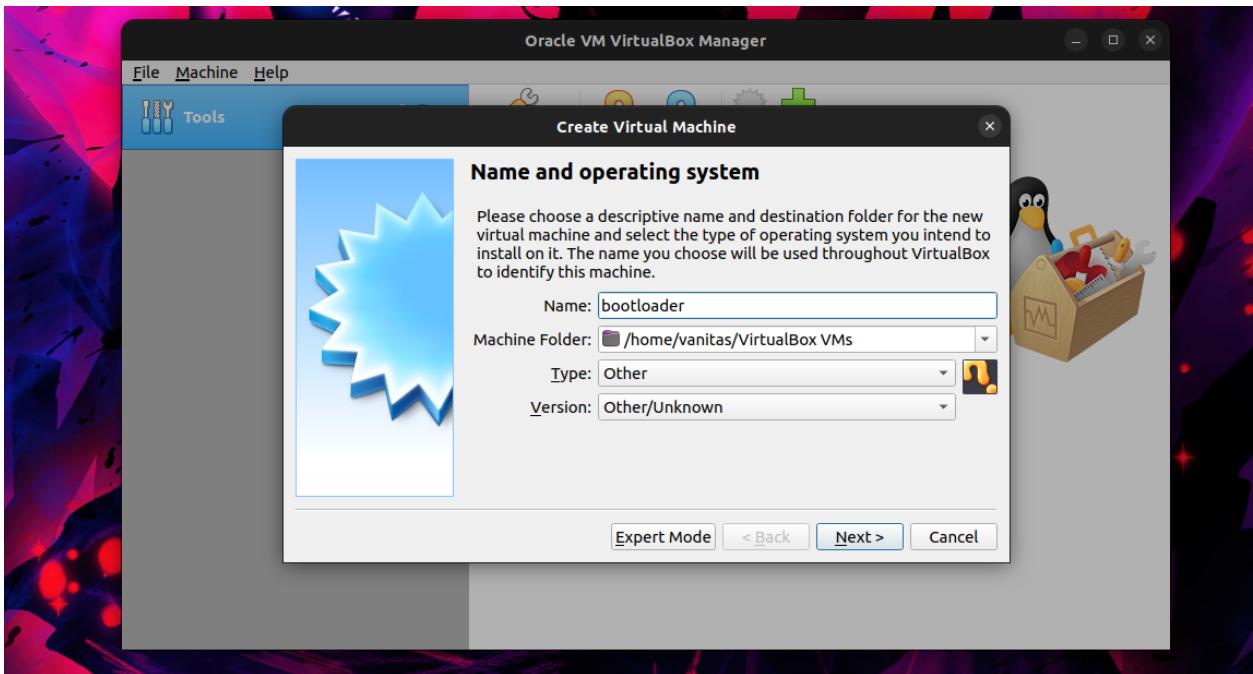


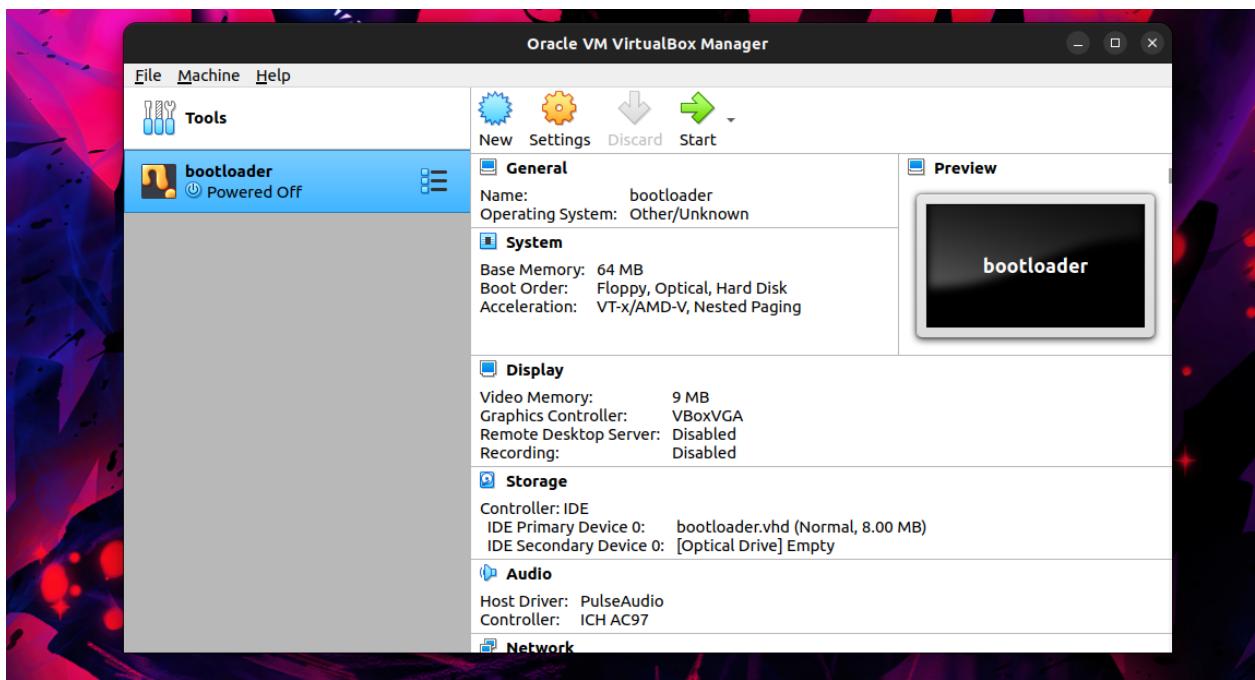
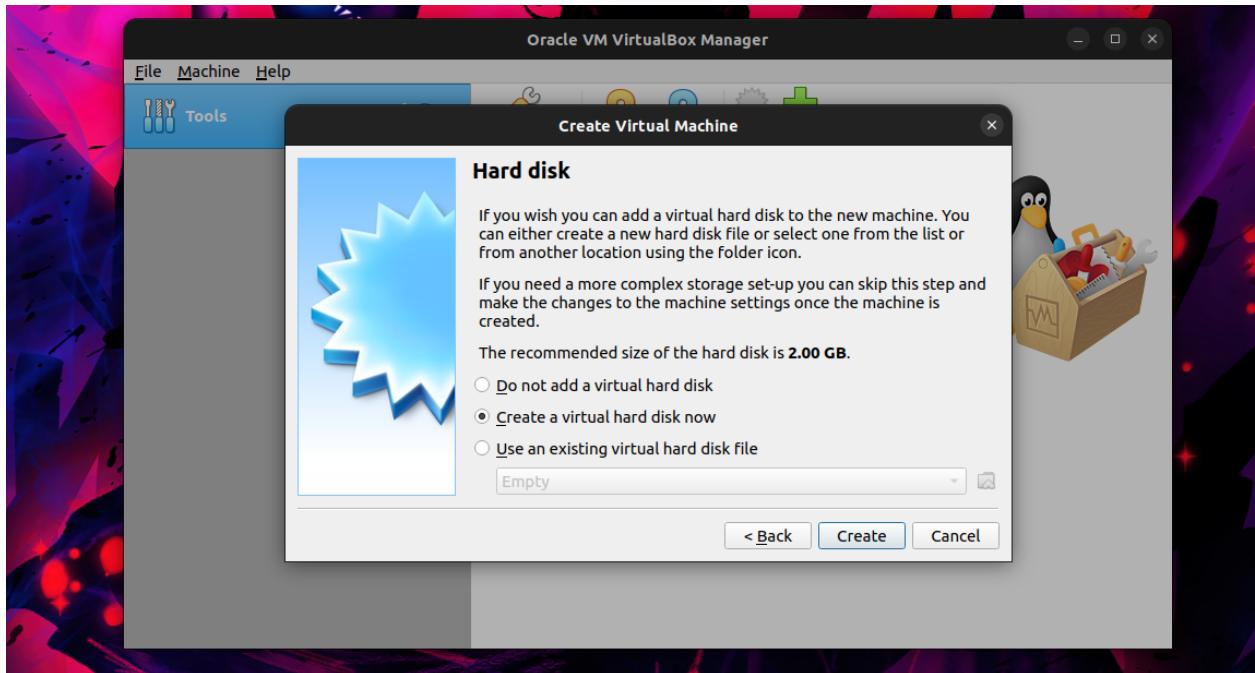
The screenshot shows a terminal window with the following command history:

```
vanitas@vinay:~/Desktop/bootloader$ gedit boot1.asm
vanitas@vinay:~/Desktop/bootloader$ gedit boot1.asm
vanitas@vinay:~/Desktop/bootloader$ nasm boot1.asm -f bin -o boot.bin
vanitas@vinay:~/Desktop/bootloader$ dd if=boot.bin bs=512 of=outputboot.img
1+0 records in
1+0 records out
512 bytes copied, 6.0289e-05 s, 8.5 MB/s
vanitas@vinay:~/Desktop/bootloader$ S
```

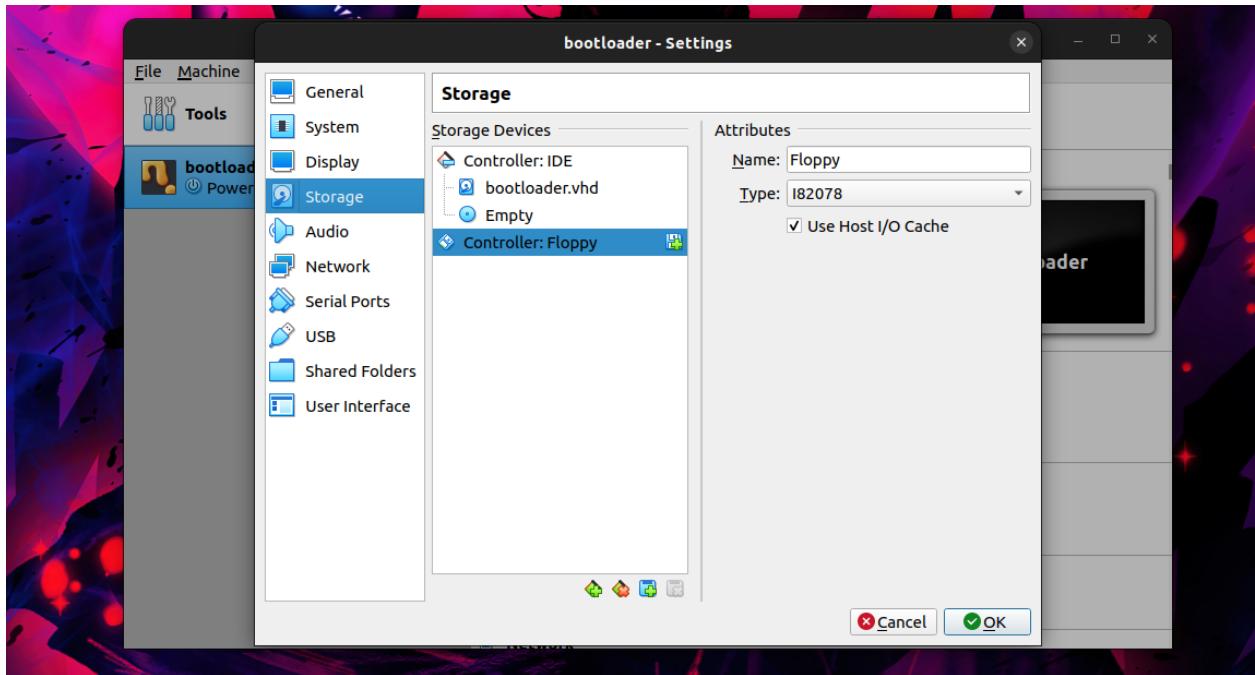
The terminal window also displays the command prompt "vanitas@vinay: ~/Desktop/bootloader\$".

Start the virtual box to then create a virtual machine in it.

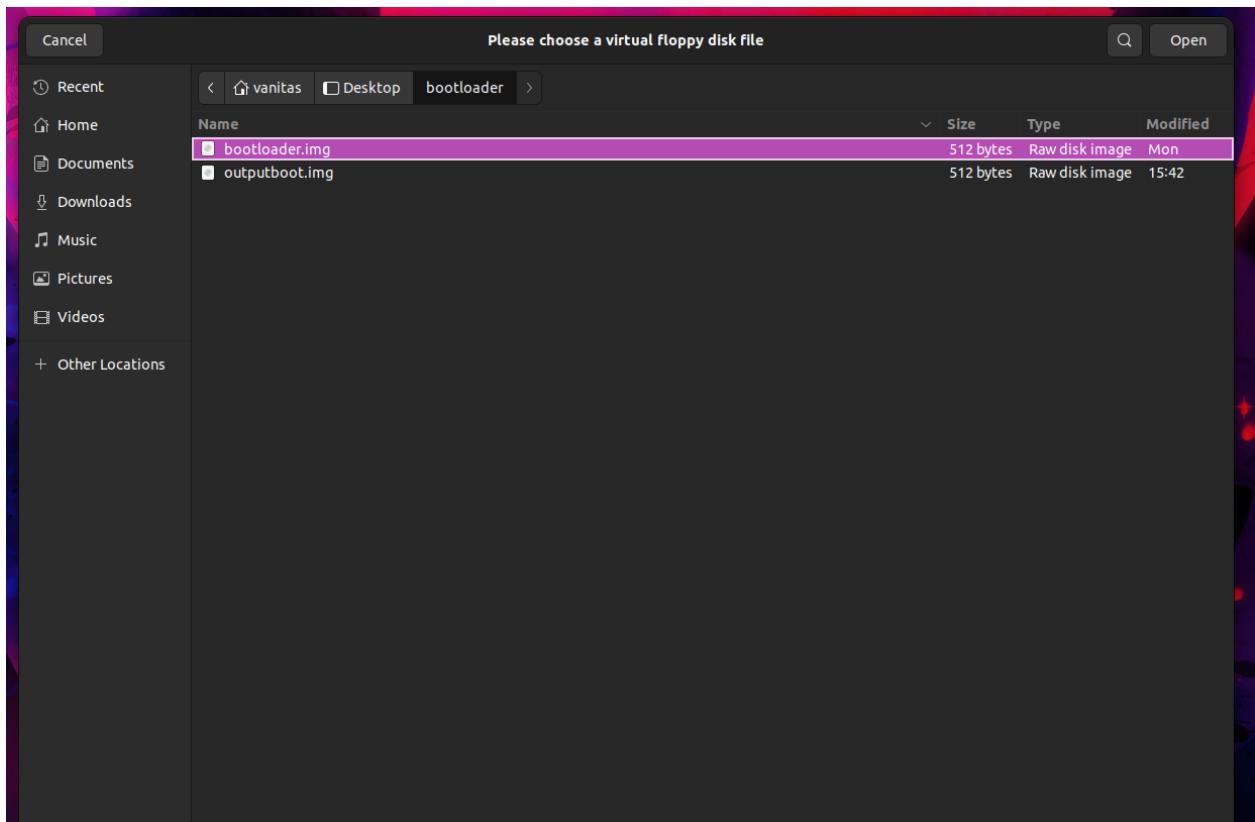


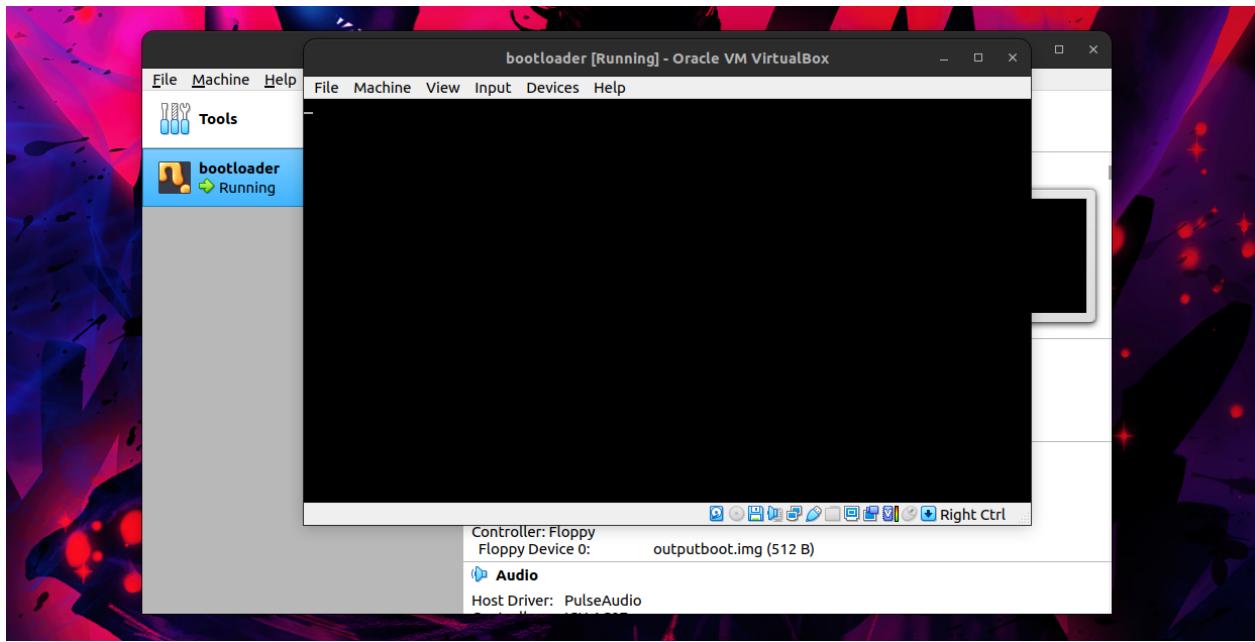


Click on settings > storage and add a floppy to the device.



Add the img file in the floppy.

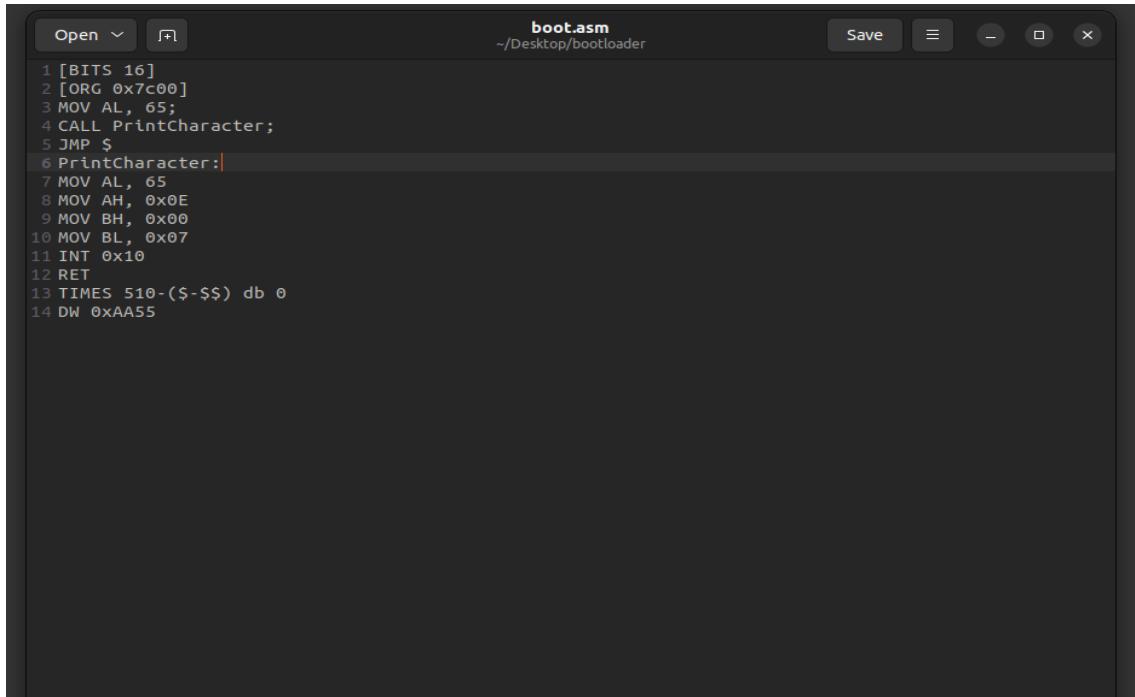




Now run the VM, you should see an empty screen with a blinking cursor.

## Printing alphabet in bootloader

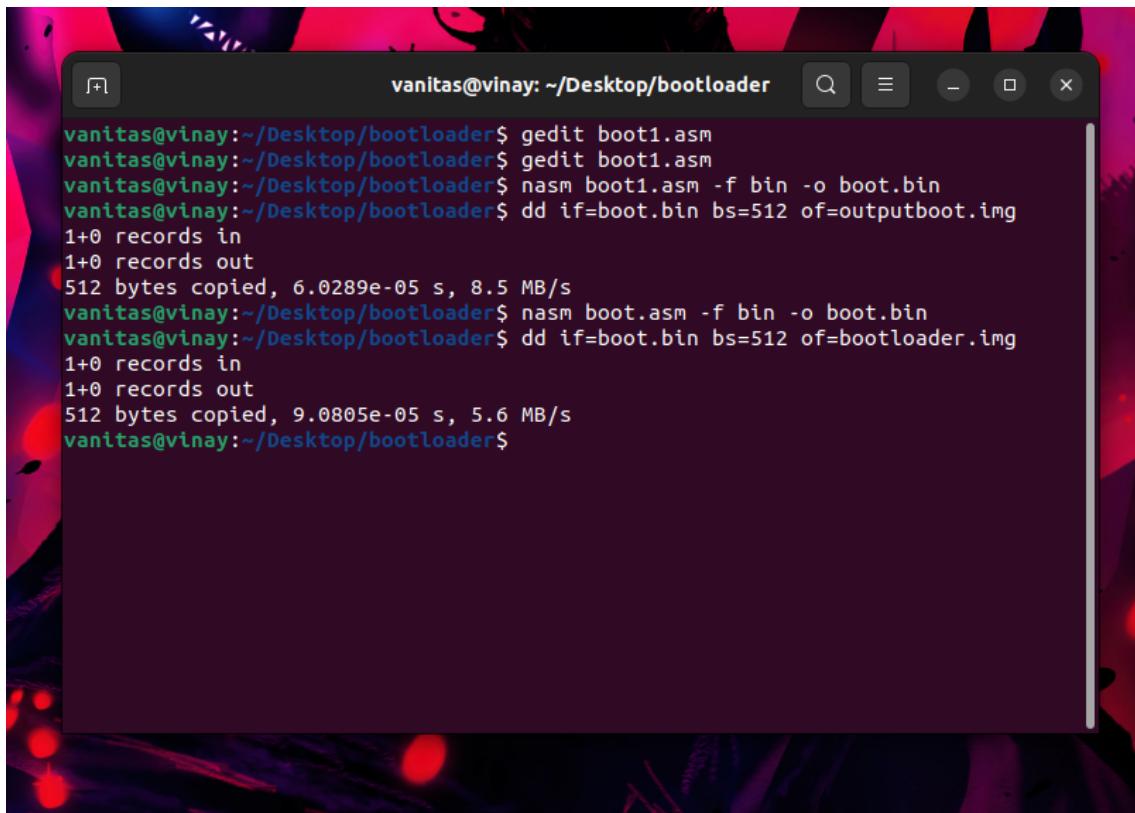
Write the assembly code in a simple text editor and save it as a ".asm" file.



A screenshot of a code editor window titled "boot.asm". The file path is shown as "~/Desktop/bootloader". The code editor has standard controls like "Open", "Save", and a search bar. The assembly code in the editor is:

```
1 [BITS 16]
2 [ORG 0x7c00]
3 MOV AL, 65;
4 CALL PrintCharacter;
5 JMP $ 
6 PrintCharacter:|
7 MOV AL, 65
8 MOV AH, 0x0E
9 MOV BH, 0x00
10 MOV BL, 0x07
11 INT 0x10
12 RET
13 TIMES 510-($-$) db 0
14 DW 0xAA55
```

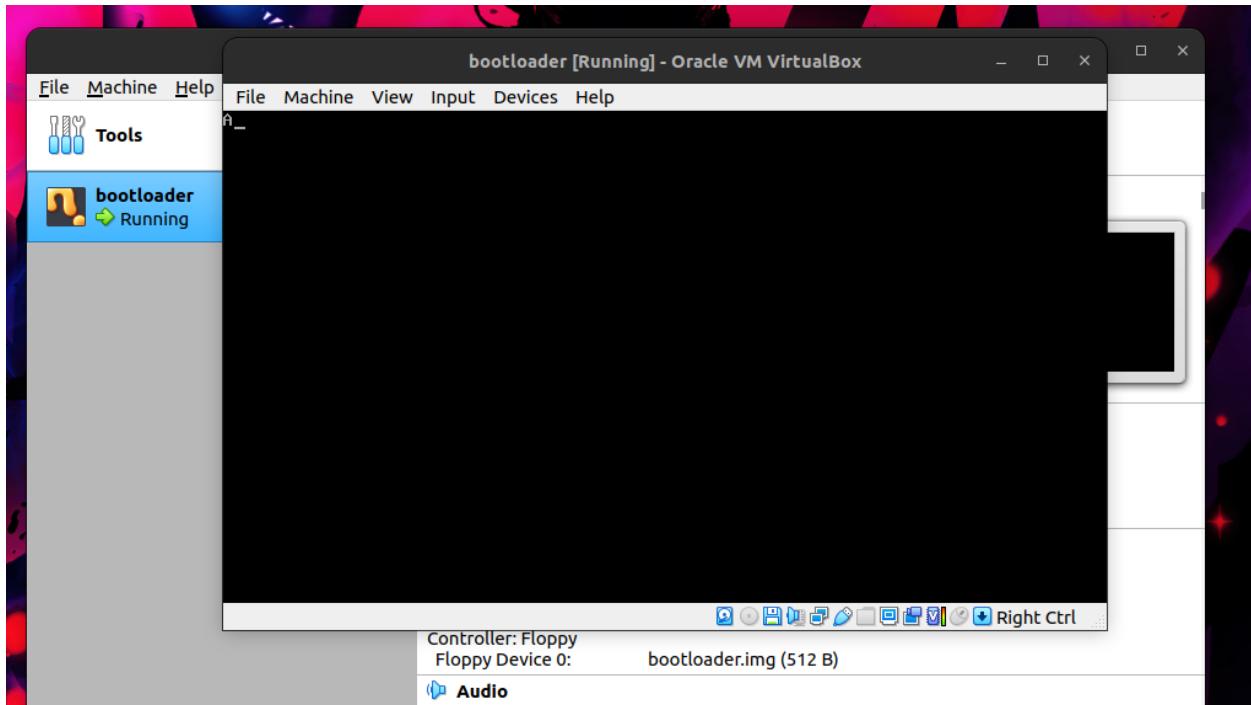
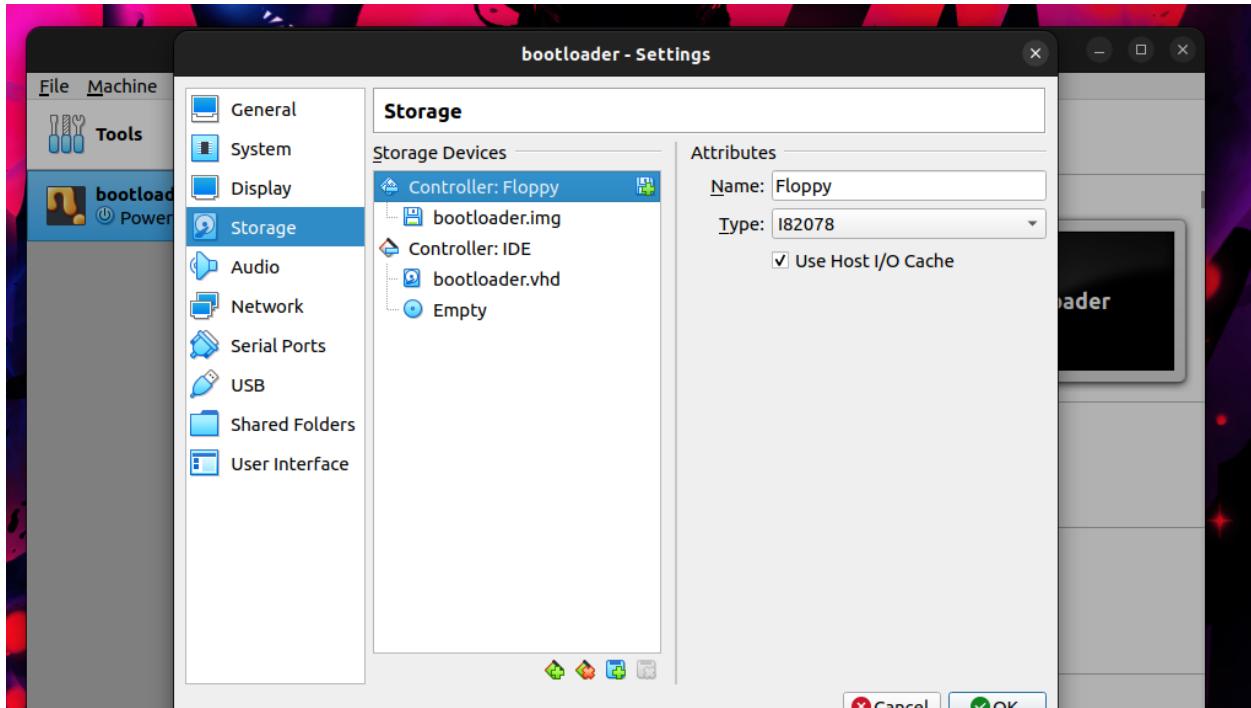
Compile the code using nasm in the Linux terminal to convert it first into a binary file and then into an iso img file.



A screenshot of a terminal window titled "vanitas@vinay: ~/Desktop/bootloader". The terminal shows the following command sequence and its output:

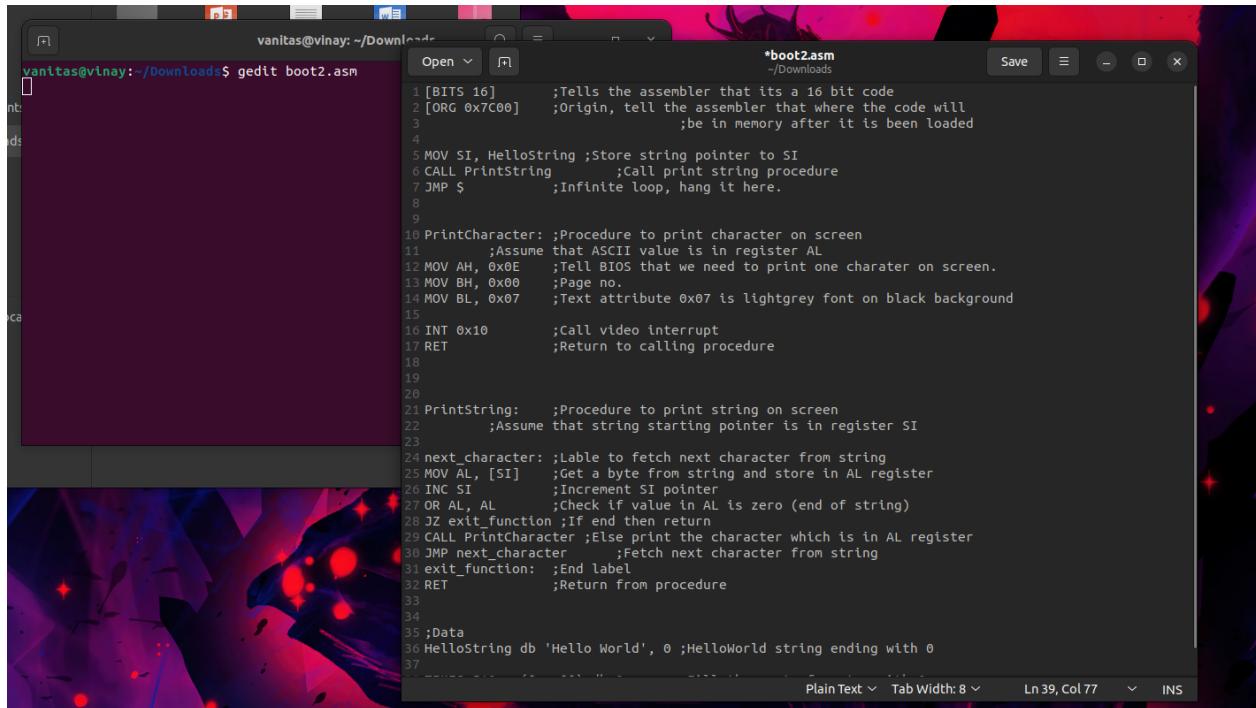
```
vanitas@vinay:~/Desktop/bootloader$ gedit boot1.asm
vanitas@vinay:~/Desktop/bootloader$ gedit boot1.asm
vanitas@vinay:~/Desktop/bootloader$ nasm boot1.asm -f bin -o boot.bin
vanitas@vinay:~/Desktop/bootloader$ dd if=boot.bin bs=512 of=outputboot.img
1+0 records in
1+0 records out
512 bytes copied, 6.0289e-05 s, 8.5 MB/s
vanitas@vinay:~/Desktop/bootloader$ nasm boot.asm -f bin -o boot.bin
vanitas@vinay:~/Desktop/bootloader$ dd if=boot.bin bs=512 of=bootloader.img
1+0 records in
1+0 records out
512 bytes copied, 9.0805e-05 s, 5.6 MB/s
vanitas@vinay:~/Desktop/bootloader$
```

Start the virtual box to then create a virtual machine in it, and repeat the same steps.



## Displaying “Hello World” in bootloader:

Write the assembly code in a simple text editor and save it as a “.asm” file.



The screenshot shows a terminal window titled "vanitas@vinay: ~/Downloads\$ gedit boot2.asm". The code is an assembly program named "boot2.asm". The code defines a 16-bit bootloader that prints "Hello World" to the screen. It includes procedures for printing characters and strings, and handles the INT 0x10 interrupt for video output. The assembly code is as follows:

```
1 [BITS 16]      ;Tells the assembler that its a 16 bit code
2 [ORG 0x7C00]    ;Origin, tell the assembler that where the code will
                  ;be in memory after it is been loaded
3
4
5 MOV SI, HelloString ;Store string pointer to SI
6 CALL PrintString   ;Call print string procedure
7 JMP $              ;Infinite loop, hang it here.
8
9
10 PrintCharacter: ;Procedure to print character on screen
11     ;Assume that ASCII value is in register AL
12 MOV AH, 0x0E       ;Tell BIOS that we need to print one character on screen.
13 MOV BH, 0x00       ;Page no.
14 MOV BL, 0x07       ;Text attribute 0x07 is lightgrey font on black background
15
16 INT 0x10          ;Call video interrupt
17 RET               ;Return to calling procedure
18
19
20
21 PrintString:     ;Procedure to print string on screen
22     ;Assume that string starting pointer is in register SI
23
24 next_character:  ;Label to fetch next character from string
25 MOV AL, [SI]       ;Get a byte from string and store in AL register
26 INC SI             ;Increment SI pointer
27 OR AL, AL          ;Check if value in AL is zero (end of string)
28 JZ exit_function  ;If end then return
29 CALL PrintCharacter ;Else print the character which is in AL register
30 JMP next_character ;Fetch next character from string
31 exit_function:    ;End label
32 RET               ;Return from procedure
33
34
35 ;Data
36 HelloString db 'Hello World', 0 ;HelloWorld string ending with 0
37
```

Code:

[BITS 16] ;Tells the assembler that its a 16 bit code

[ORG 0x7C00] ;Origin, tell the assembler that where the code will  
;be in memory after it is been loaded

MOV SI, HelloString ;Store string pointer to SI

CALL PrintString ;Call print string procedure

JMP \$ ;Infinite loop, hang it here.

PrintCharacter: ;Procedure to print character on screen

;Assume that ASCII value is in register AL

MOV AH, 0x0E ;Tell BIOS that we need to print one character on screen.

```
MOV BH, 0x00      ;Page no.  
MOV BL, 0x07;Text attribute 0x07 is lightgrey font on black background
```

```
INT 0x10      ;Call video interrupt  
RET          ;Return to calling procedure
```

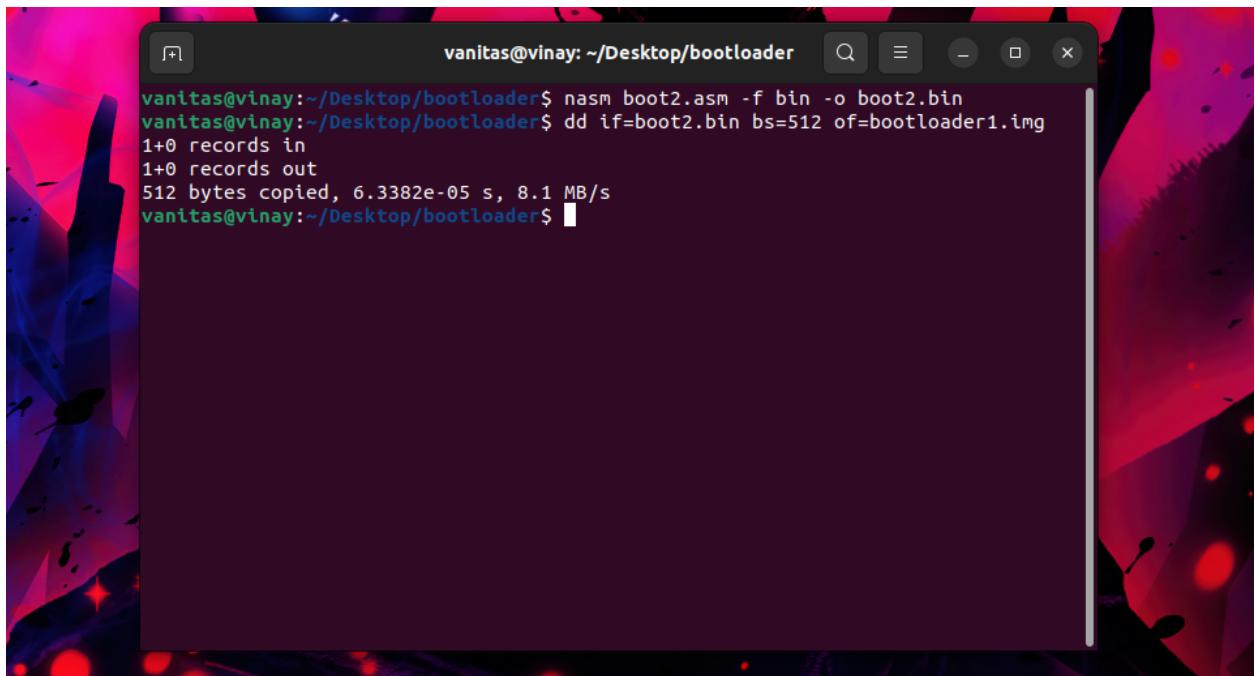
```
PrintString:    ;Procedure to print string on screen  
               ;Assume that string starting pointer is in register SI
```

```
next_character:    ;Label to fetch next character from string  
MOV AL, [SI]  ;Get a byte from string and store in AL register  
INC SI        ;Increment SI pointer  
OR AL, AL     ;Check if value in AL is zero (end of string)  
JZ exit_function ;If end then return  
CALL PrintCharacter ;Else print the character which is in AL register  
JMP next_character  ;Fetch next character from string  
exit_function:   ;End label  
RET          ;Return from procedure
```

```
;Data  
HelloString db 'Hello World', 0      ;HelloWorld string ending with 0
```

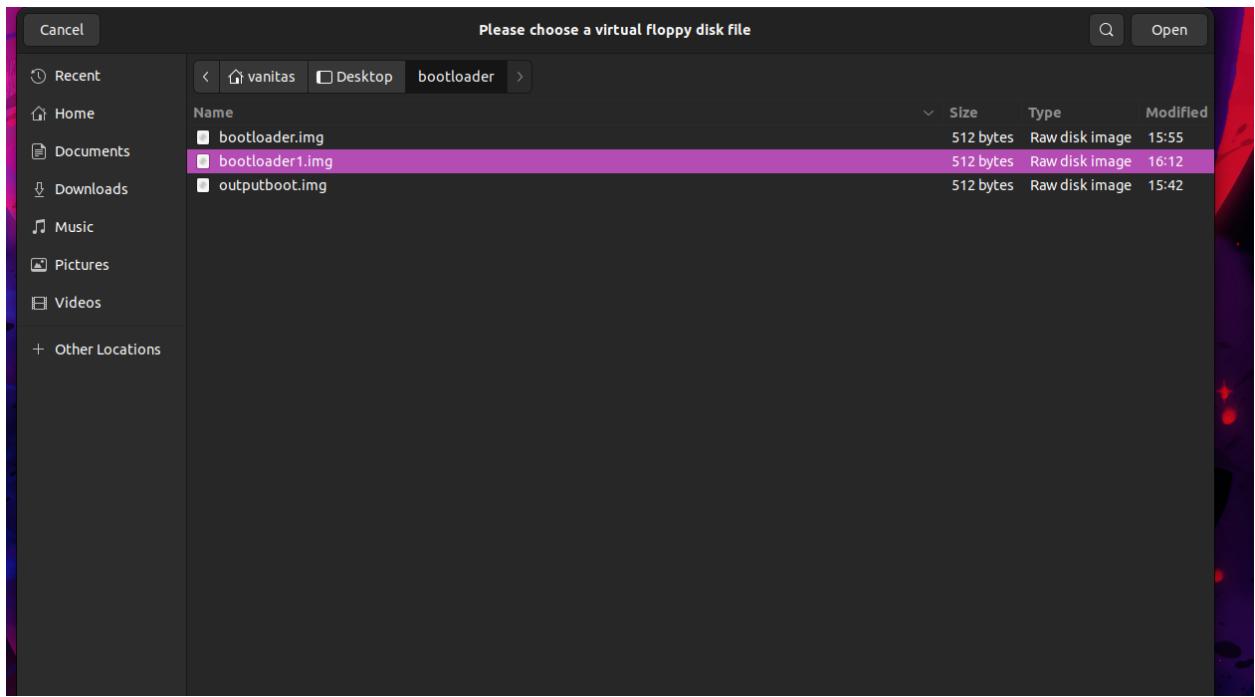
```
TIMES 510 - ($ - $$) db 0      ;Fill the rest of sector with 0  
DW 0xAA55                      ;Add boot signature at the end of bootloader
```

Compile the code using nasm in the Linux terminal to convert it first into a binary file and then into an iso img file.



```
vanitas@vinay:~/Desktop/bootloader$ nasm boot2.asm -f bin -o boot2.bin
vanitas@vinay:~/Desktop/bootloader$ dd if=boot2.bin bs=512 of=bootloader1.img
1+0 records in
1+0 records out
512 bytes copied, 6.3382e-05 s, 8.1 MB/s
vanitas@vinay:~/Desktop/bootloader$
```

Start the virtual box to then create a virtual machine in it, and repeat the same steps.



We can see “Hello World” printed.

