**BOOTLOADER PROGRAM**

1. **Create a first Bootloader that does nothing**

create a file firstBootLoader.asm

[BITS 16] ;tell 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

JMP $ ;infinite loop

TIMES 510 - ($ - $$) db 0 ;fill the rest of sector with 0

DW 0xAA55 ; add boot signature at the end of bootloader

Explanation:

[BITS 16]: Our code starts with [BITS 16] which is an assembler directive. This will tell assembler that our code is a 16 bit code.

[ORG 0x7C00]: Then we have used [ORG 0x7C00] which tell assembler to assemble the instructions from Origin 0x7C00. BIOS loads bootloader at physical address 0x7C00 hence we have assemble our bootloader starting from that location.

JMP $: JMP at location $ means jumping to the same location. Thus this nothing but an infinite loop. We just want to hang our code here.

TIMES 510 - ($ - $$) db 0: As bootloader is always of length 512 bytes, our code does not fit in this size as its small. We need to use rest of memory and hence we clear it out using TIMES directive. $ stands for start of instruction and $$ stands for start of program. Thus ($ - $$) means length of our code.

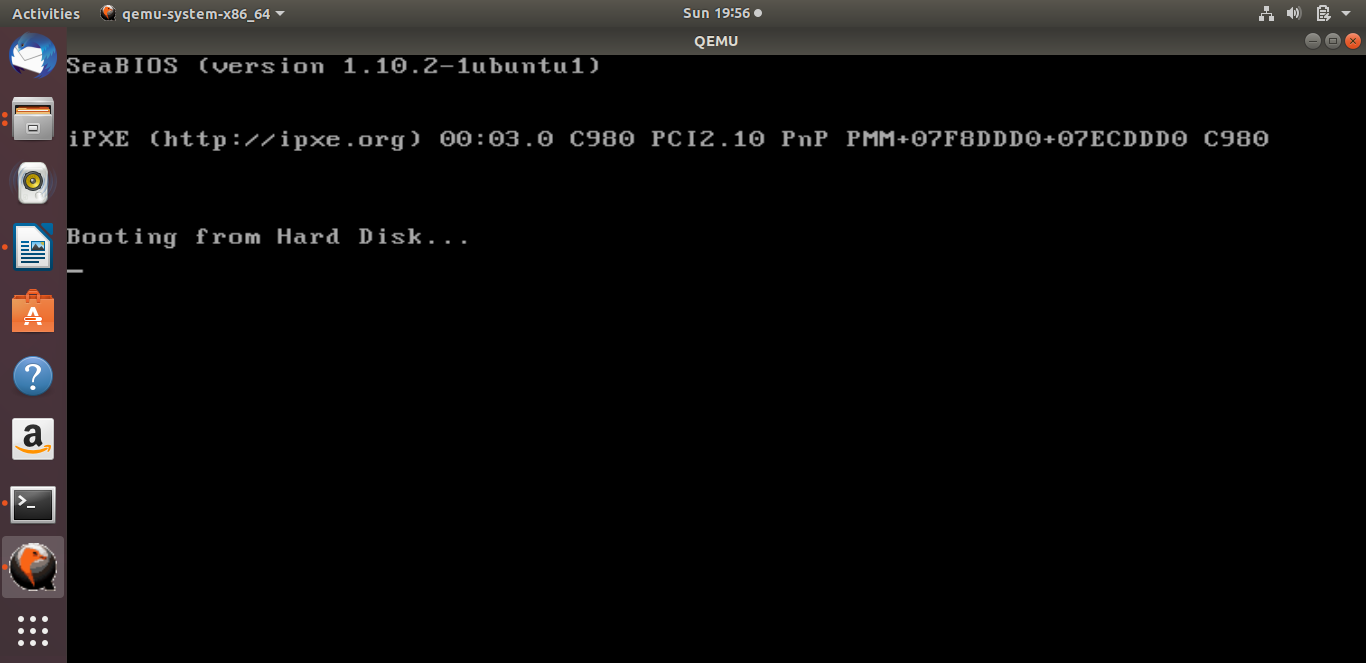
DW 0xAA55: This is boot signature. This tells the BIOS that this is a valid bootloader. If bios does not get 0x55 and 0xAA at the end of the bootloader than it will treat bootloader as invalid. Thus we provide this two bytes at the end of our bootloader.

Compile the program

nasm firstBootLoader.asm -f bin -o boot.bin

Run using the emulator

qemu-system-x86\_64 -drive file=boot.bin,index=0,media=disk,format=raw



1. **Create 2nd Bootloader that prints ‘A’ on the screen**

we will use BIOS video interrupt int 0x10.

INT 0x10 is a BIOS video interrupt. All the video related calls are made through this interrupt.

To use this interrupt we need to set the values of some register.

AL = ASCII value of character to display

AH = 0x0E ;Teletype mode (This will tell bios that we want to print one character on screen)

BL = Text Attribute (This will be the fore ground and background color

of character to be displayed. 0x07 in our case.)

BH = Page Number (0x00 for most of the cases)

Code saved in secondBootLoader.asm

*bits 16*

*org 0x7c00*

*boot:*

*mov si,hello*

*mov ah, 0x0e*

*.loop:*

*lodsb*

*or al,al*

*jz halt*

*int 0x10*

*jmp .loop*

*halt:*

*cli*

*hlt*

*hello: db "A",0*

*times 510 - ($-$$) db 0*

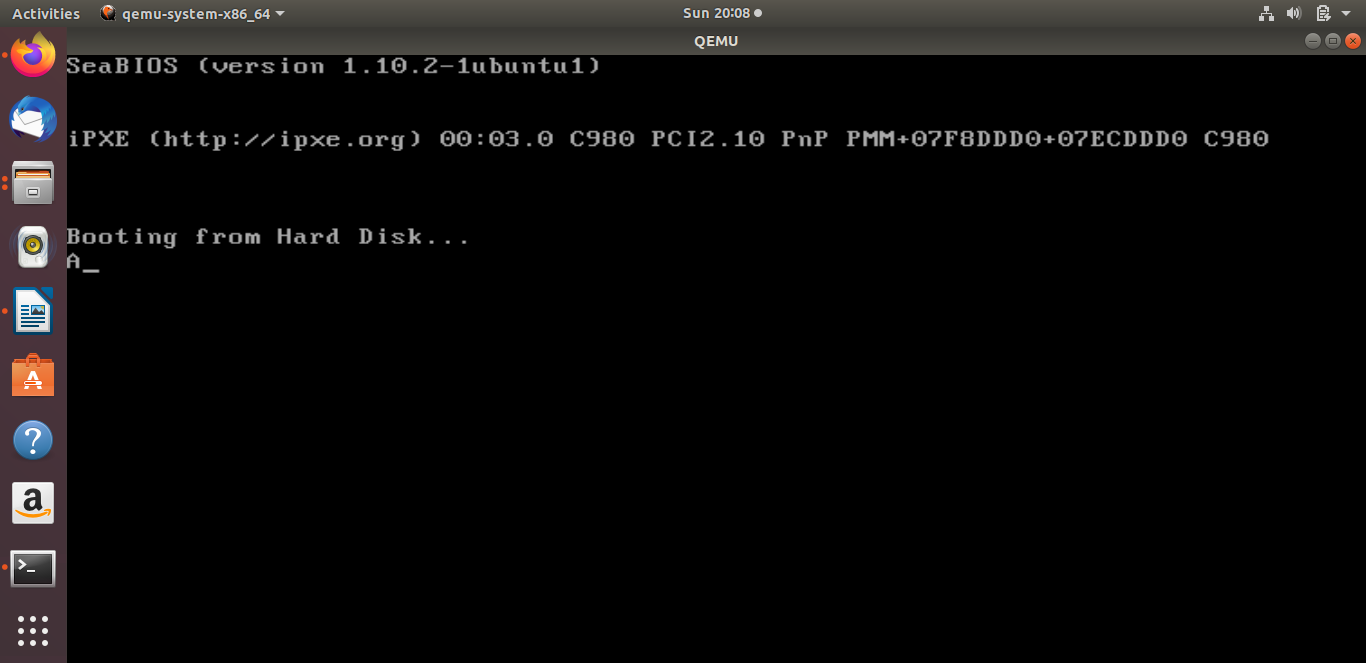
*dw 0xaa55*

Compile the program

nasm secondBootLoader.asm -f bin -o boot2.bin

Run using the emulator

*qemu-system-x86\_64 -drive file=boot2.bin,index=0,media=disk,format=raw*



Try this too.

**Code saved in secBootLoader2.asm( Another code for secondboot loader)**

[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 AL, 65

CALL PrintCharacter

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 charater 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

TIMES 510 - ($ - $$) db 0 ;Fill the rest of sector with 0

DW 0xAA55 ;Add boot signature at the end of bootloader

1. **Third Bootloader that prints your name say “Shyamala”**

Its time to create our final YourName bootloader. We have enough experience now and can code it without wasting a second. So once again start your favorite text editor and start writing following code

