Загрузчик Hello World.

Для того чтобы разобраться как работает загрузчик, создадим простой загрузчик Hello World.

Чтобы загрузчик работал должны выполняться следующие правила:

1. Загрузчик должен занимать ровно 512 байт.
2. Загрузчик должен заканчиваться кодом 55hAAh.

Где загрузчик располагается в памяти:

По адресу 7C00h

Int 10h:

Это прерывание BIOSa для управлением экраном.

Как им пользоваться:

AH — 0x0E AL -ASCII символ который отображается

BH — номер видео страницы (обычно 0x00)

BL — текстовый атрибут(обычно 0x07)

Затем вызываем прерывание:

Int 10h

Установка загрузчика

1. Сохранить загрузчик как boot.asm
2. Откомпилировать: nasm boot.asm
3. Убедиться что длина 512 байт.
4. Записать загрузчик на первый сектор диска.

В linux:

dd if=загрузчик bs=512 of=/dev/диск (куда пишется загрузчик, например: sdb1, sdc1

sudo dd if=boot2 bs=512 of=/dev/sdb1

Как определить имена дисков в Linux - sudo fdisk -l

Создание загрузчика.

1 попытка
Hanging Bootloader
Этот загрузчик ничего не делает просто распределяет память и виснет.

```
[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
```

Вторая попытка:Print a character Bootloader

INT 0x10 это BIOS прерывание экрана.

AL = ASCII ;значение символа для отображения.
AH = 0x0E ;режим вывода посимвольный
BL = Text Attribute Цвет fore ground и background отображаемого символа. 0x07 в нашем случае
BH = Номер видео страницы (0x00 для большинства случаев)

После заполнения регистров вызываем прерывание.

```
[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 afterit 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
```

```
[BITS 16]          ; 16 bit code generation
[ORG 0x7C00]       ; ORGin location is 7C00
;Main program
main:              ; Main program label
mov ah,0x0E        ; This number is the number of the
function in the BIOS to run.
                   ;  This function is put character on
screen function
mov bh,0x00        ; Page number (I'm not 100% sure of this
myself but it is best
                   ;  to leave it as zero for most of the
work we will be doing)
mov bl,0x07        ; Text attribute (Controls the background
and foreground colour
                   ;  and possibly some other options)
                   ;  07 = White text, black background.
                   ; (Feel free to play with this value as it
shouldn't harm
                   ;  anything)
mov al,65          ; This should (in theory) put a ASCII
value into al to be
                   ;  displayed. (This is not the normal way
to do this)
int 0x10           ; Call the BIOS video interrupt.
```

```
jmp $                ; Put it into a coninuous loop to stop it
running off into
                     ;  the memory running any junk it may find
there.
; End matter
times 510-($-$$) db 0    ; Fill the rest of the sector with
zeros
dw 0xAA55                ; Boot signature
```

Hello World! -загрузчик

```
[BITS 16]       ; 16 bit code generation
[ORG 0x7C00]    ; Origin location
; Main program
main:           ; Label for the start of the main
program
 mov ax,0x0000  ; Setup the Data Segment register
                ; Location of data is DS:Offset
 mov ds,ax      ; This can not be loaded directly
it has to be in two steps.
                ; 'mov ds, 0x0000' will NOT work
due to limitations on the CPU
 mov si, HelloWorld     ; Load the string into
position for the procedure.
 call PutStr    ; Call/start the procedure
jmp $           ; Never ending loop
```

```
; Procedures
PutStr:          ; Procedure label/start
 ; Set up the registers for the interrupt call
 mov ah,0x0E     ; The function to display a character
(teletype)
 mov bh,0x00     ; Page number
 mov bl,0x07     ; Normal text attribute
```

```
.nextchar          ; Internal label (needed to loop round
for the next character)
 lodsb             ; I think of this as LOaD String Block
                   ; (Not sure if thats the real meaning
though)
                   ; Loads [SI] into AL and increases SI
by one
 ; Check for end of string '0'
 or al,al          ; Sets the zero flag if al = 0
                   ; (OR outputs 0's where there is a
zero bit in the register)
 jz .return        ; If the zero flag has been set go to
the end of the procedure.
                   ; Zero flag gets set when an
instruction returns 0 as the answer.
 int 0x10          ; Run the BIOS video interrupt
 jmp .nextchar     ; Loop back round tothe top
.return            ; Label at the end to jump to when
complete
 ret               ; Return to main program
```

```
; Data
HelloWorld db 'Hello World',13,10,0
; End Matter
times 510-($-$$) db 0   ; Fill the rest with zeros
dw 0xAA55               ; Boot loader signature
```

Компилируем загрузчик в простой bin формат:
(для windows:
nasmw boot.asm -f bin -o boot.bin)
Для Ubuntu:
nasm -f bin -o boot.bin boot.asm

Создаём образ floppy диска:
dd if=/dev/zero of=floppy.img bs=1024 count=1440

Копируем загрузчик в начальный сектор образа floppy диска:
dd if=boot.bin of=floppy.img seek=0 count=1 conv=notrunc

Создаём папку iso

```
mkdir iso
```

Копируем в неё образ:

```
cp floppy.img iso/
```

Создаём iso образ:

```
genisoimage -quiet -V 'MYOS'
-input-charset iso8859-1 -o myos.iso -b
floppy.img -hide floppy.img iso/
```

Создаём виртуальную машину в Virtual Box и

Загружаем iso образ как cd привод.