

Daniels NASM Bootstraps Tutorial

Daniel Marjamäki

Preface

This tutorial is a guide for those who want to create their own bootstraps.

The Basics

These are the rules that you must follow:

- The BIOS will load your bootstrap to address 07C00h. Sadly, the segment and offset varies.
- Bootstraps must be compiled as plain binary files.
- The filesize for the plain binary file must be 512 bytes.
- The file must end with AA55h.

A Minimal Bootstrap

This bootstrap just hangs:

```
; HANG.ASM
; A minimal bootstrap

hang:                ; Hang!
                    jmp hang

times 510-($-$$) db 0 ; Fill the file with 0's
dw 0AA55h            ; End the file with AA55
```

The line starting with "times" is a command that only NASM understands. The line will insert 0's until the filesize is 510 bytes. The whole file will therefore be 512 bytes. The last instruction puts AA55 at the end of the file.

To compile the bootstrap, use this command:

```
nasm hang.asm -o hang.bin
```

If you want to test the bootstrap, you must first put it on the first sector on a floppy disk. You can for example use 'dd' or 'rawrite'. When the bootstrap is on the floppy, test it by restarting your computer with the floppy inserted. The computer should hang then.

The Memory Problem

There is a memory problem. As I've written bootstraps are always loaded to address 07C00. We don't know what segment and offset the BIOS has put us in. The segment can be anything between 0000 and 07C0. This is a problem when we want to use variables. The solution is simple. Begin your bootstrap by jumping to your bootstrap, but jump to a known segment.

Here is an example:

```
; JUMP.ASM
; Make a jump and then hang

; Tell the compiler that this is offset 0.
; It isn't offset 0, but it will be after the jump.
[ORG 0]

                    jmp 07C0h:start          ; Goto segment 07C0
```

Daniels NASM Bootstraps Tutorial

Daniel Marjamäki

```
start:
    ; Update the segment registers
    mov ax, cs
    mov ds, ax
    mov es, ax
hang:      ; Hang!
    jmp hang

times 510-($-$$) db 0
dw 0AA55h
```

If you compile and test this bootstrap, there will be no visible difference to the minimal bootstrap presented earlier. The computer will just hang.

Some Exercises

1. Create a bootstrap that outputs "====" on the screen, and then hangs. Tip: modify the jump.asm program.
2. Create a bootstrap that outputs "Hello Cyberspace!" and hangs.
3. Create a bootstrap that loads a program off the floppy disk and jumps to it.

Solutions to the Exercises

1.

```
; 1.ASM
; Print "====" on the screen and hang

; Tell the compiler that this is offset 0.
; It isn't offset 0, but it will be after the jump.
[ORG 0]
    jmp 07C0h:start      ; Goto segment 07C0

start:
    ; Update the segment registers
    mov ax, cs
    mov ds, ax
    mov es, ax

    mov ah, 9           ; Print "===="
    mov al, '='         ;
    mov bx, 7           ;
    mov cx, 4           ;
    int 10h             ;

hang:      ; Hang!
    jmp hang

times 510-($-$$) db 0
dw 0AA55h
```

2.

```
; 2.ASM
```

Daniels NASM Bootstraps Tutorial

Daniel Marjamäki

```
; Print "Hello Cyberspace!" on the screen and hang

; Tell the compiler that this is offset 0.
; It isn't offset 0, but it will be after the jump.
[ORG 0]
    jmp 07C0h:start      ; Goto segment 07C0

; Declare the string that will be printed
msg     db 'Hello Cyberspace!'

start:
    ; Update the segment registers
    mov ax, cs
    mov ds, ax
    mov es, ax

    mov si, msg          ; Print msg
print:
    lodsb                ; AL=memory contents at DS:SI

    cmp al, 0            ; If AL=0 then hang
    je hang

    mov ah, 0Eh          ; Print AL
    mov bx, 7
    int 10h

    jmp print            ; Print next character

hang:
    jmp hang             ; Hang!

times 510-($-$$) db 0
dw 0AA55h
```

3.

```
; 3.ASM
; Load a program off the disk and jump to it

; Tell the compiler that this is offset 0.
; It isn't offset 0, but it will be after the jump.
[ORG 0]
    jmp 07C0h:start      ; Goto segment 07C0

start:
    ; Update the segment registers
    mov ax, cs
    mov ds, ax
    mov es, ax

reset:
    ; Reset the floppy drive
    mov ax, 0            ;
    mov dl, 0            ; Drive=0 (=A)
    int 13h              ;
    jc reset             ; ERROR => reset again
```

Daniels NASM Bootstraps Tutorial

Daniel Marjamäki

```
read:
    mov ax, 1000h        ; ES:BX = 1000:0000
    mov es, ax          ;
    mov bx, 0           ;

    mov ah, 2           ; Load disk data to ES:BX
    mov al, 5           ; Load 5 sectors
    mov ch, 0           ; Cylinder=0
    mov cl, 2           ; Sector=2
    mov dh, 0           ; Head=0
    mov dl, 0           ; Drive=0
    int 13h             ; Read!

    jc read             ; ERROR => Try again

    jmp 1000h:0000      ; Jump to the program

times 510-($-$$) db 0
dw 0AA55h
```

This is a small loadable program.

```
; PROG.ASM

    mov ah, 9
    mov al, '='
    mov bx, 7
    mov cx, 10
    int 10h

hang:
    jmp hang
```

This program creates a disk image file that contains both the bootstrap and the small loadable program.

```
; IMAGE.ASM
; Disk image

%include '3.asm'
%include 'prog.asm'
```

Finally

Thanks for reading. Email me any suggestions, comments, questions, ... If you don't use NASM and are having problems with the code, you should contact me. Together we can solve it.