CSCI315 – Operating Systems Design

Department of Computer Science
Bucknell University

Booting an Operating System

External reading

This set of notes is based on notes from the textbook authors, as well as L. Felipe Perrone and other instructors. Xiannong Meng, Fall 2021.

Steps in Booting an OS (1)

- Power on
- The BIOS (Basic I/O System) does the following
  - Power On Self Test (POST) to check all hardware
  - Find out the “booting drive” name
    - It is configurable through BIOS, e.g., USB, CD-ROM, hard-disk …
  - Read and execute the program at the boot sector from the booting drive (a 512 byte program)
    - This program is usually called the **bootstrap program**.
    - It is written in assembly. (why?)

Steps in Booting an OS (2)

- The bootstrap program does a number of things
  - Set up registers so that memory can be separated into segments such as code, data, heap, and stack.
  - Initialize register values so that proper memory addresses can be computed.
  - Set up the partition table for the booting device.
  - Read the rest of the operating system program into memory from the booting device and start running the operating system.

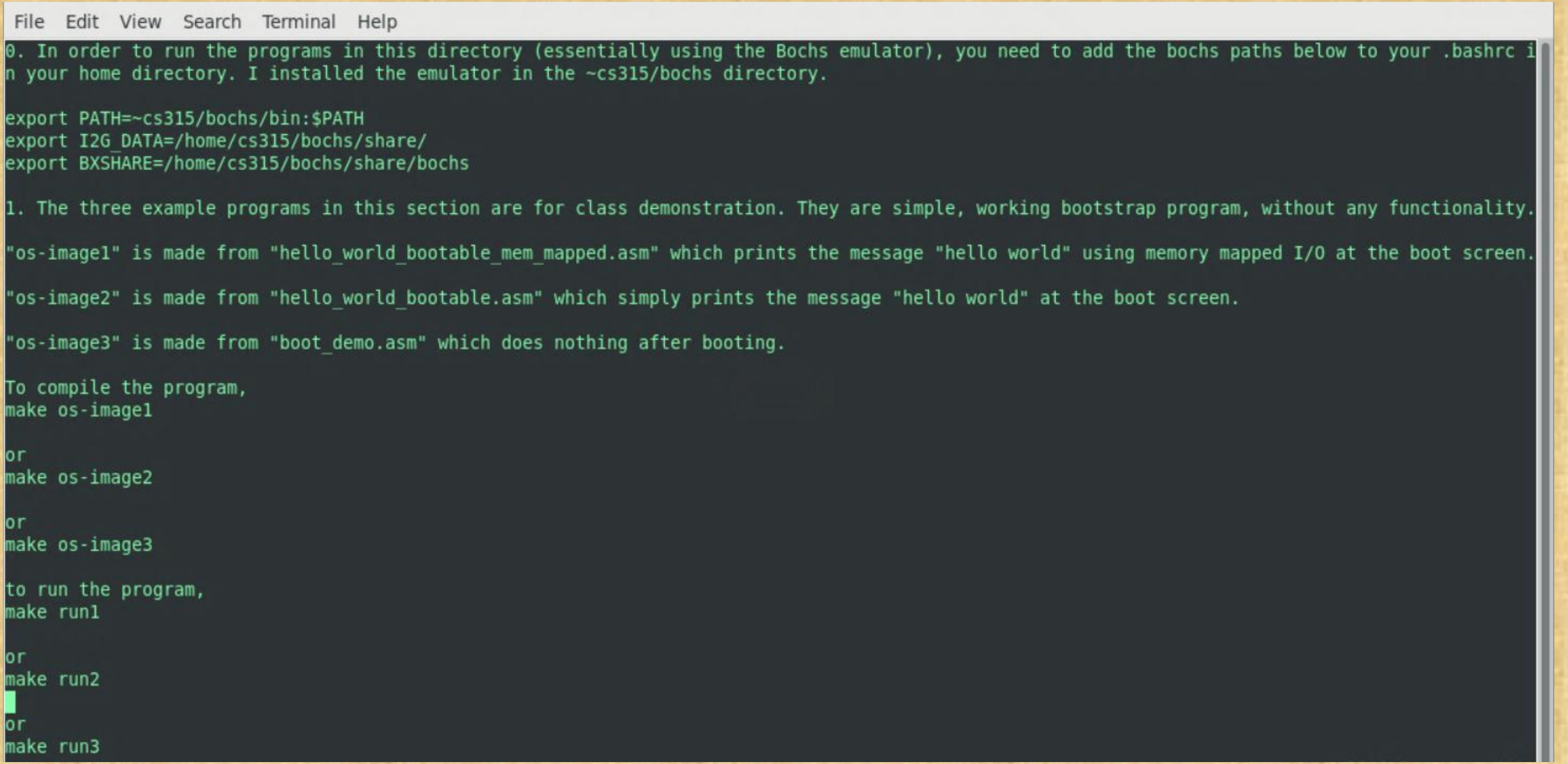
Examining Two Bootstrap Examples

- Let’s look at two simple, working, but incomplete bootstrap examples.
  - “boot_demo.asm” which does nothing at the end of the startup.
  - “hello_world_bootable_mem_mapped.asm” which prints a “hello world” message through a memory-mapped buffer.

Executing the Examples

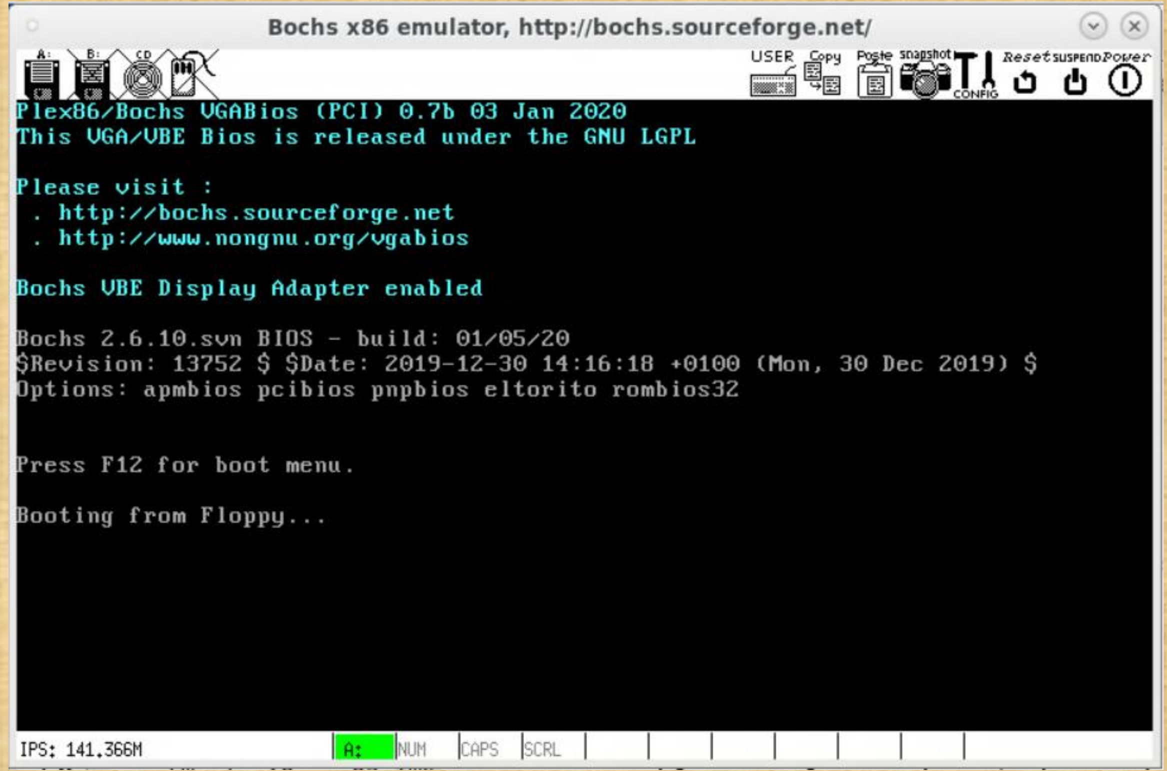
- These programs can run on a real x86 machine or on an emulator.
- We’ll run them on an emulator, **bochs**. The following are the screenshots of these two programs in execution.
- Read the “readme.txt” and “Makefile” in the directory of http://www.eg.bucknell.edu/~cs315/F2021/meng/code/boot for details.
- Go to the directory, compile, and run the program.

Compile and Run Bochs Emulator



```
0. In order to run the programs in this directory (essentially using the Bochs emulator), you need to add the bochs paths below to your .bashrc i
n your home directory. I installed the emulator in the ~cs315/bochs directory.

export PATH=~cs315/bochs/bin:$PATH
export I2G_DATA=/home/cs315/bochs/share/
export BXSHARE=/home/cs315/bochs/share/bochs

1. The three example programs in this section are for class demonstration. They are simple, working bootstrap program, without any functionality.

"os-image1" is made from "hello_world_bootable_mem_mapped.asm" which prints the message "hello world" using memory mapped I/O at the boot screen.

"os-image2" is made from "hello_world_bootable.asm" which simply prints the message "hello world" at the boot screen.

"os-image3" is made from "boot_demo.asm" which does nothing after booting.

To compile the program,
make os-image1

or
make os-image2

or
make os-image3

to run the program,
make run1

or
make run2

or
make run3
```

“boot_demo” that does nothing

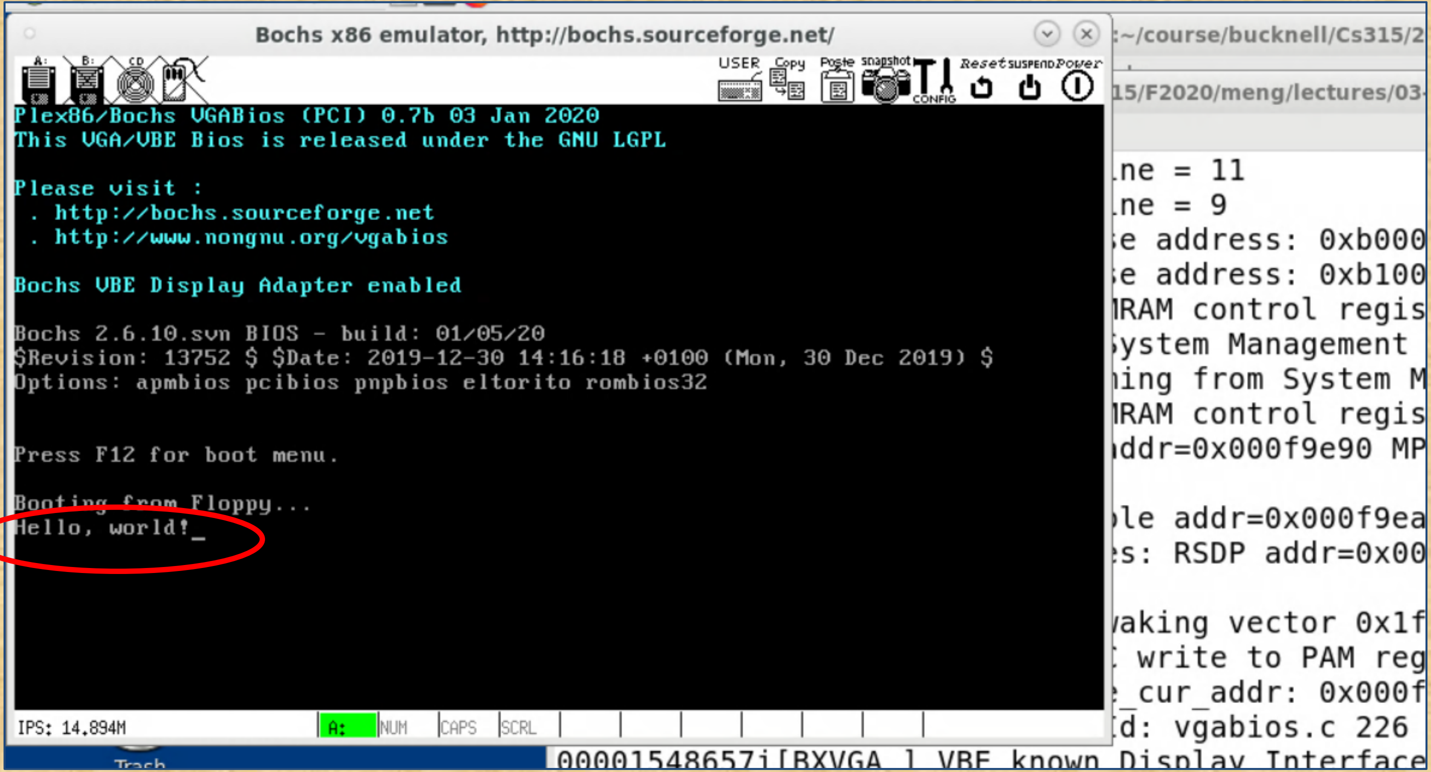


```
Plex86/Bochs VGABios (PCI) 0.7b 03 Jan 2020
This VGA/VBE Bios is released under the GNU LGPL

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


Press F12 for boot menu.

Booting from Floppy...
```

http://www.eg.bucknell.edu/~cs315/F2021/meng/code/boot/boot_demo.asm

https://www.cs.bham.ac.uk/~exr/lectures/opsys/10_11/lectures/os-dev.pdf

“hello_world_bootable”


Bochs x86 emulator, http://bochs.sourceforge.net/
A:
B:
CD
USER
Copy
Paste
snapshot
CONFIG
Reset
SUSPEND
Power
IPS: 14.894M
A:
NUM
CAPS
SCRL
:~/course/bucknell/Cs315/2
15/F2020/meng/lectures/03
Trash

```
Plex86/Bochs 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


Press F12 for boot menu.

Booting from Floppy...
Hello, world!_
```

```
.ne = 11
.ne = 9
;e address: 0xb000
;e address: 0xb100
IRAM control regis
;ystem Management
ning from System M
IRAM control regis
iddr=0x000f9e90 MP

ole addr=0x000f9ea
es: RSDP addr=0x00

vaking vector 0x1f
 write to PAM reg
e_cur_addr: 0x000f
Id: vgabios.c 226
00001548657i[BXVGA ] VBE known Display Interface
```

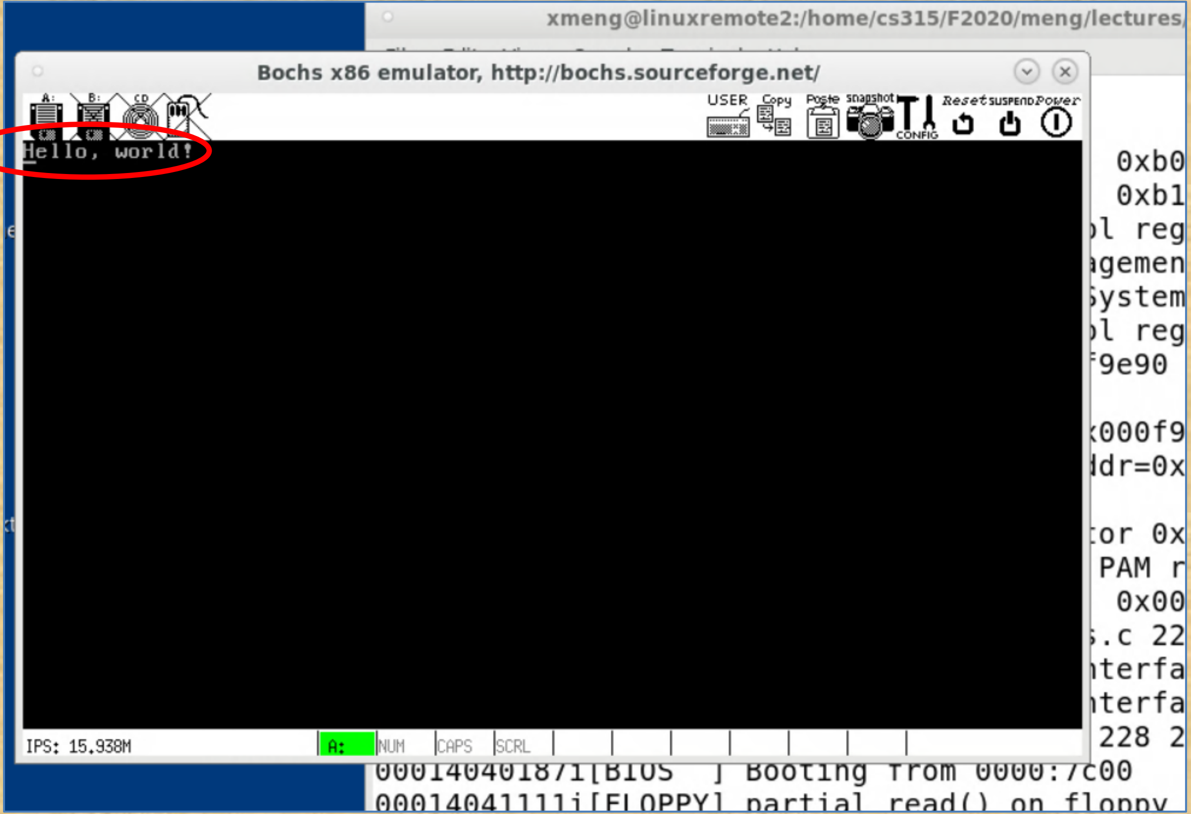
Printed by the bootstrap program

http://www.eg.bucknell.edu/~cs315/F2021/meng/code/boot/hello_world_bootable.asm

Original from: https://thiscouldbebetter.wordpress.com/2011/03/15/creating-a-bootable-program-in-assembly-language/

"hello_world_mem_mapped"

Printed by our bootstrap program, screen cleared.


xmeng@linuxremote2:/home/cs315/F2020/meng/lectures/
Bochs x86 emulator, http://bochs.sourceforge.net/
USER
Copy
Paste
snapshot
CONFIG
Reset
SUSPEND
Power
IPS: 15.938M
A:
NUM
CAPS
SCRL

```
Hello, world!
```

```
 0xb0
 0xb1
ol reg
agemen
System
ol reg
f9e90

x000f9
ddr=0x

tor 0x
 PAM r
 0x00
s.c 22
nterfa
nterfa
 228 2
00014040187i[BIOS  ] Booting from 0000:7c00
00014041111i[FLOPPY] partial read() on floppy
```

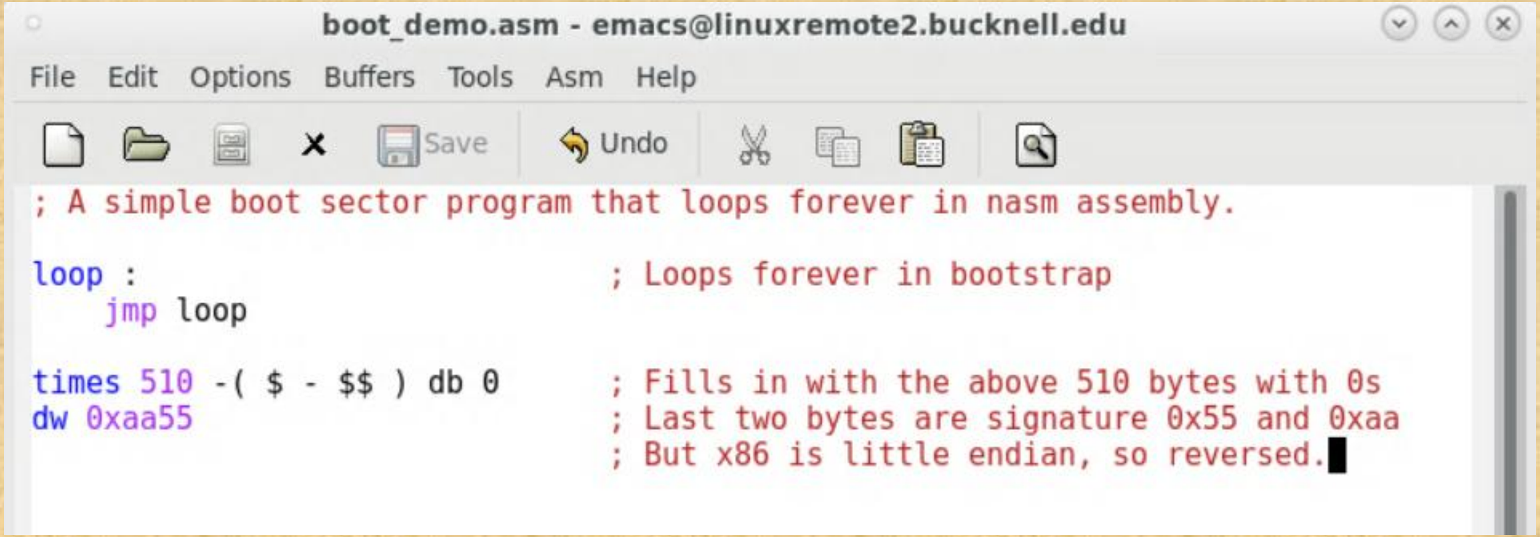
http://www.eg.bucknell.edu/~cs315/F2021/meng/code/boot/hello_world_bootable_mem_mapped.asm

Original from: https://thiscouldbebetter.wordpress.com/2011/03/17/displaying-text-in-assembly-without-interrupts/

Four Key Features

1. The program must reside at the boot sector of the booting device, e.g., a disk, a CD, or a USB drive.
2. The size of the program must be 512 bytes.
3. The 511th byte must have the value of **`0x55`** and 512th bytes **`0xAA`**.
4. The program must be in an infinite loop.

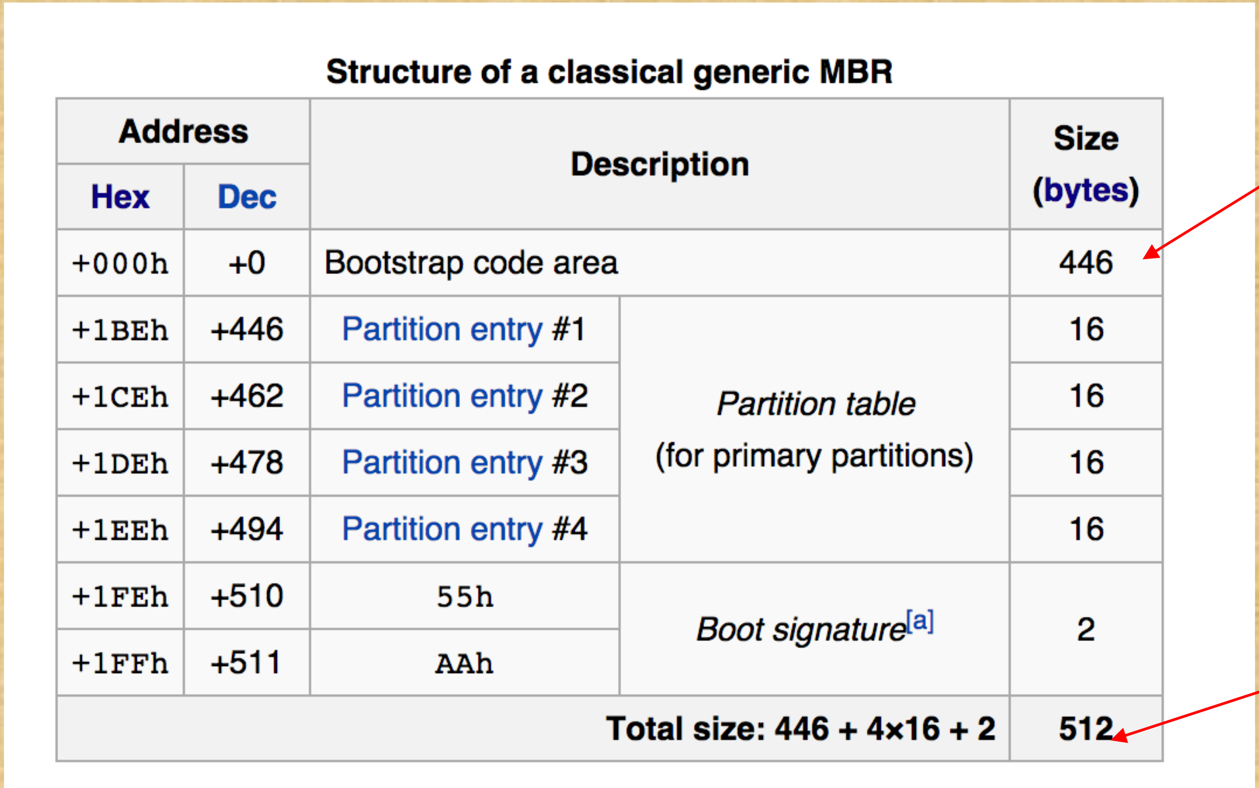
Simplest Bootstrap Program


boot_demo.asm - emacs@linuxremote2.bucknell.edu
File Edit Options Buffers Tools Asm Help
Save
Undo

```
; A simple boot sector program that loops forever in nasm assembly.

loop :                          ; Loops forever in bootstrap
    jmp loop

times 510 -( $ - $$ ) db 0      ; Fills in with the above 510 bytes with 0s
dw 0xaa55                       ; Last two bytes are signature 0x55 and 0xaa
                                ; But x86 is little endian, so reversed.
```

boot_demo.asm

Logic Structure of Boot Loader



Structure of a classical generic MBR

Address		Description		Size (bytes)
Hex	Dec			
+000h	+0	Bootstrap code area		446
+1BEh	+446	Partition entry #1	*Partition table* (for primary partitions)	16
+1CEh	+462	Partition entry #2		16
+1DEh	+478	Partition entry #3		16
+1EEh	+494	Partition entry #4		16
+1FEh	+510	55h	*Boot signature*[a]	2
+1FFh	+511	AAh		
			Total size: 446 + 4×16 + 2	**512**

code size

total size

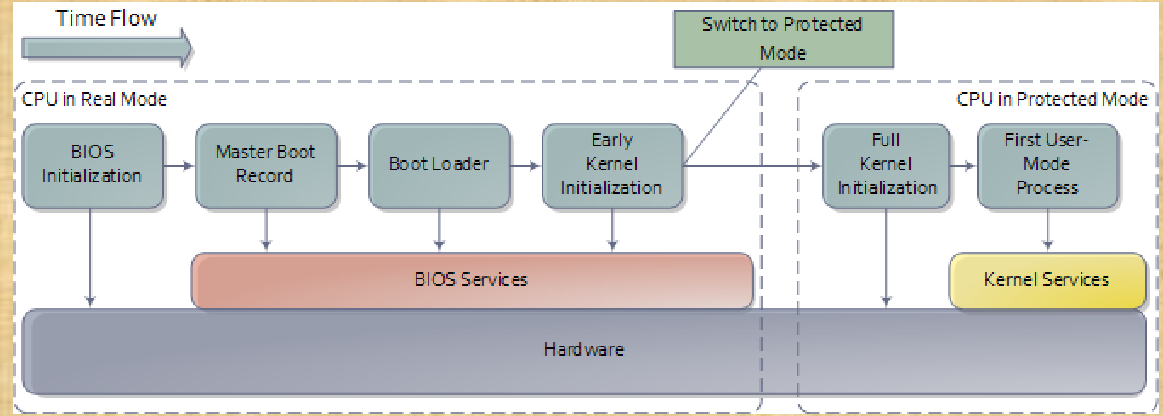
MBR: Master Boot Record

Partition Table Entry

- Each of the four partition table entry is a 16-byte value, indicating how the device (disk) is partitioned.

https://en.wikipedia.org/wiki/Master_boot_record#PTE

Booting up the OS



- BIOS is firmware (flash memory). Power on self tests (POST) check if machine is in shape to run.
- Every bootable disk has an MBR, which contains a bootstrap program and a partition table. Each partition has a boot sector with the boot loader.

Source: http://duartes.org/gustavo/blog/post/how-computers-boot-up/