How to turn on your computer*

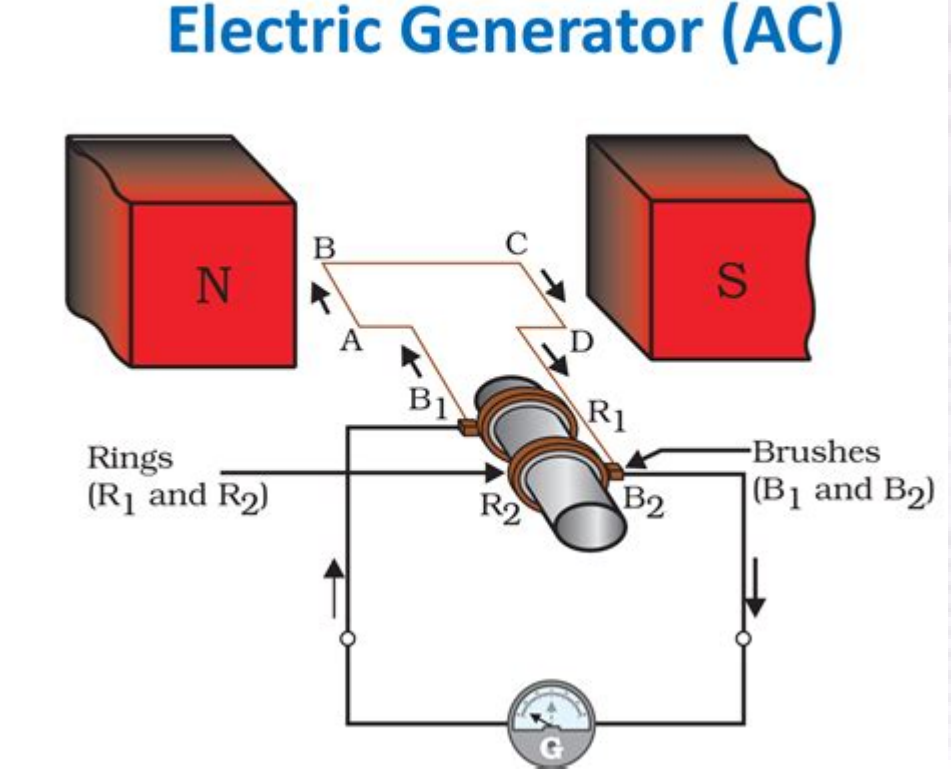
* x86 computer **

** old x86 computer

Step one: invent the wheel

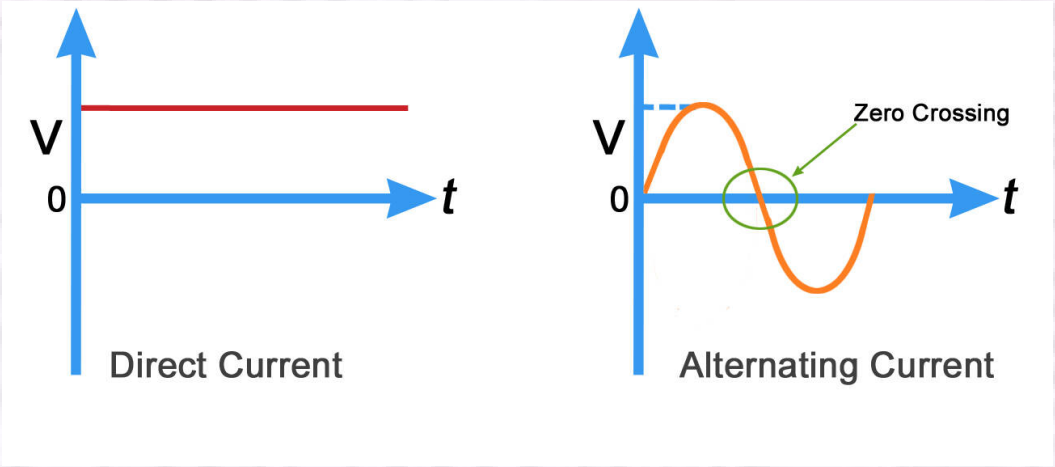
- Changing magnetic flux through a loop induces a current in the loop
- V α dϕ/dt
- Basically how every form of energy production (coal, nuclear, wind, geothermal) is converted into electricity

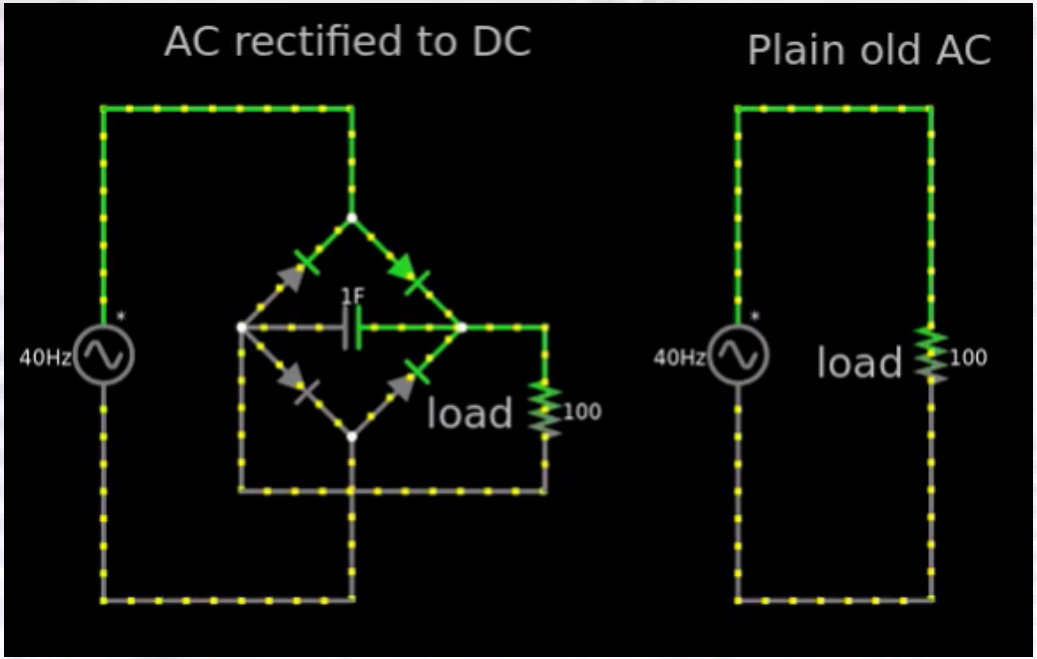
  (solar is an exception to this)



Step two: rectify AC to DC

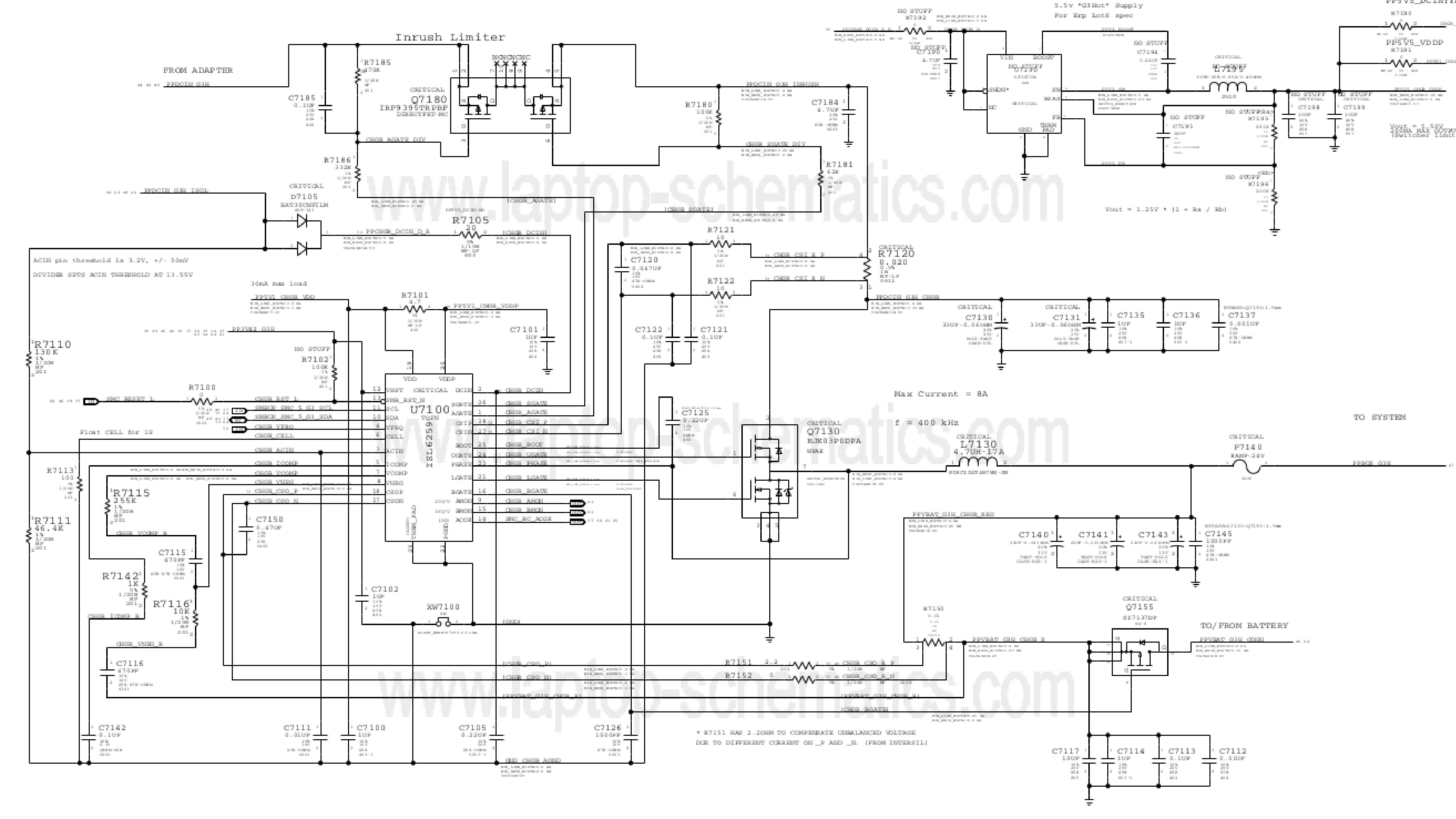
- AC bad for computer! Computer hate AC. AC probably make CPU go boom.
- DC good for computer. DC smooth like pebble. Computer love DC.
- Filters, stepdown transformers, grounding are also important



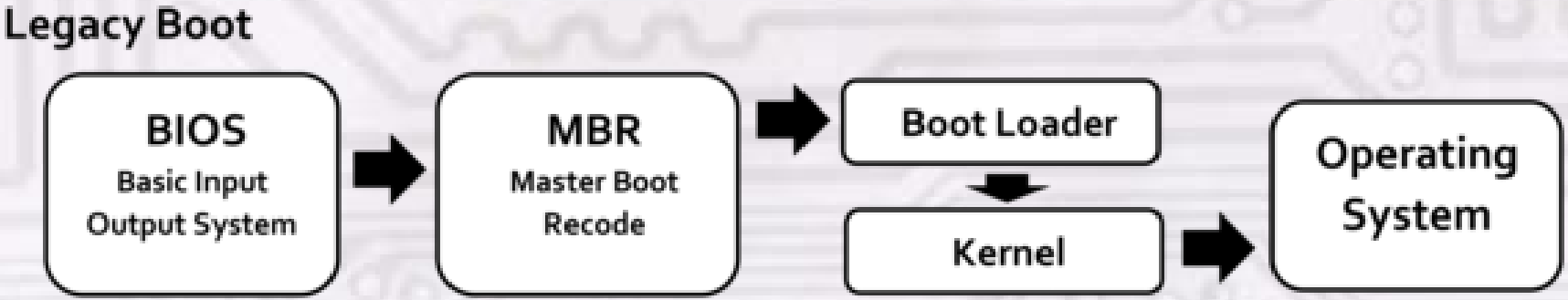


Step three: plug it in!

- Some protective circuitry on motherboard, lots of chips that do onboard logic
  - Regulating voltage, generating data / clock signals, monitoring power
  - Distinction between power lines and data lines
- POST (power-on self test)
  - Ensuring that onboard hardware is all working properly
  - Example, many computers will beep if no RAM sticks are found
  - (Therefore they fail to POST)

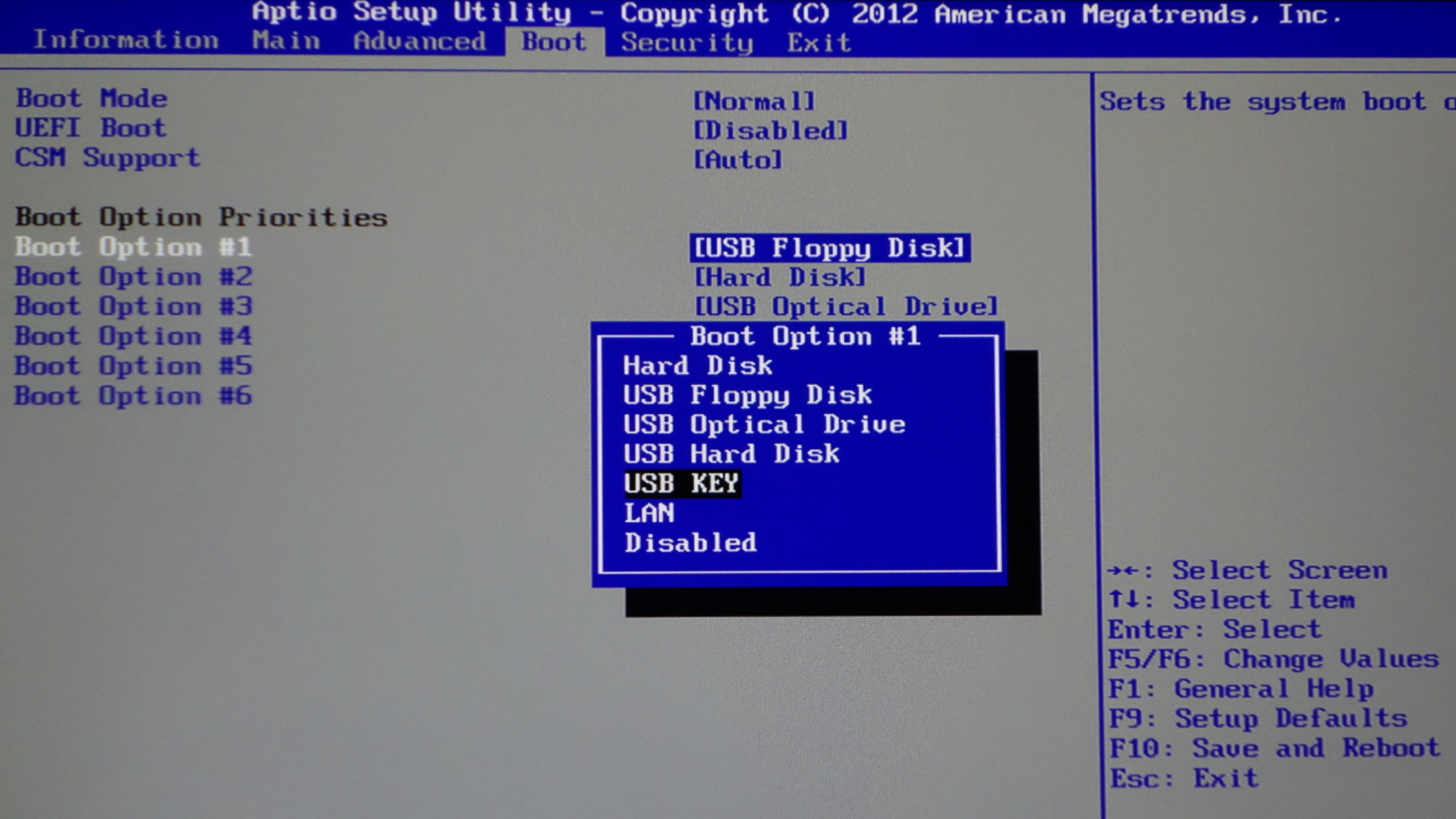

Inrush Limiter
FROM ADAPTER
PPDCIN_G3H
PPDCIN_G3H_ISOL
PPDCIN_G3H_INRUSH
CHGR_AGATE_DIV
CHGR_SGATE_DIV
CRITICAL
Q7180
IRF9395TRPBF
DIRECTFET-MC
R7185
R7186
R7180
R7181
C7185
C7184
D7105
BAT30CMFILM
R7105
PPCHGR_DCIN_D_R
ACIN pin threshold is 3.2V, +/- 50mV
DIVIDER SETS ACIN THRESHOLD AT 13.55V
30mA max load
PP5V1_CHGR_VDD
PP5V1_CHGR_VDDP
PP3V42_G3H
R7101
R7102
NO STUFF
R7110
130K
R7111
46.4K
R7100
SMC_RESET_L
CHGR_RST_L
SMBUS_SMC_5_G3_SCL
SMBUS_SMC_5_G3_SDA
CHGR_VFRQ
CHGR_CELL
CHGR_ACIN
CHGR_ICOMP
CHGR_VCOMP
CHGR_VNEG
CHGR_CSO_P
CHGR_CSO_N
Float CELL for 1S
U7100
ISL6259
TQFN
CHGR_DCIN
CHGR_SGATE
CHGR_AGATE
CHGR_CSI_P
CHGR_CSI_N
CHGR_BOOT
CHGR_UGATE
CHGR_PHASE
CHGR_LGATE
CHGR_BGATE
CHGR_AMON
CHGR_BMON
SMC_BC_ACOK
R7113
R7115
255K
R7142
1K
R7116
10K
C7115
C7116
C7150
C7101
C7102
XW7100
CHGR_VCOMP_R
CHGR_ICOMP_R
CHGR_VNEG_R
R7121
R7122
C7120
C7122
C7121
C7125
R7120
0.020
CHGR_CSI_R_P
CHGR_CSI_R_N
PPDCIN_G3H_CHGR
C7130
C7131
C7135
C7136
C7137
Max Current = 8A
f = 400 kHz
Q7130
RJK03P0DPA
L7130
4.7UH-17A
F7140
8AMP-24V
PPBUS_G3H
TO SYSTEM
PPVBAT_G3H_CHGR_REG
C7140
C7141
C7143
C7145
R7150
PPVBAT_G3H_CHGR_R
Q7155
TO/FROM BATTERY
PPVBAT_G3H_CONN
R7151
R7152
C7117
C7114
C7113
C7112
C7142
C7111
C7100
C7105
C7126
GND_CHGR_AGND
* R7151 HAS 2.2OHM TO COMPENSATE UNBALANCED VOLTAGE
DUE TO DIFFERENT CURRENT ON _P AND _N (FROM INTERSIL)
5.5V "G3Hot" Supply
For Erp Lot6 spec
R7192
C7190
U7190
C7194
L7195
C7195
C7198
C7199
R7195
R7196
PP5V5_VDDP
R7190
R7191
Vout = 1.25V * (1 + Ra / Rb)
Vout = 5.50V
www.laptop-schematics.com

Overview of the rest of the process



What happens before bootloader?

- BIOS searches for bootable drives
  - USB’s, hard disks, etc.
- Search order can be modified by booting into BIOS config and changing parameters
- BIOS looks for a drive starting with a 512 byte sector ending with the magic number 0xaa55


Aptio Setup Utility - Copyright (C) 2012 American Megatrends, Inc.
Information Main Advanced Boot Security Exit
Boot Mode [Normal]
UEFI Boot [Disabled]
CSM Support [Auto]
Boot Option Priorities
Boot Option #1 [USB Floppy Disk]
Boot Option #2 [Hard Disk]
Boot Option #3 [USB Optical Drive]
Boot Option #4
Boot Option #5
Boot Option #6
Boot Option #1
Hard Disk
USB Floppy Disk
USB Optical Drive
USB Hard Disk
USB KEY
LAN
Disabled
Sets the system boot o
→←: Select Screen
↑↓: Select Item
Enter: Select
F5/F6: Change Values
F1: General Help
F9: Setup Defaults
F10: Save and Reboot
Esc: Exit


Winbond
W83697HF
144W5F211601602
PhoenixBios™
D686 BIOS
©PHOENIX 1998
0787301198
JCD1
CN14
RN21
R453
RN36
RN35
C135
R176
R434
R165

MBR (Master Boot Record)

- First physical sector on bootable storage device
- Contains code + data required to find and boot the operating system
- x86 machines start in 16 bit Real Mode (for compatibility), operating systems normally run in 32 or 64 bit Protected Mode
- In 446 bytes (A single 512 byte sector – 2 byte magic number – partition table with 4 16-byte entries), the bootloader must*
  - find a bootable partition
  - find kernel image and load it into memory
  - switch to protected 32 bit mode (64 bit is a bit more complicated (32 of them, actually))
  - (optionally) set up stack for operating system

* this is partially a lie, as we shall see

MASTER BOOT RECORD

INVOKE-IR

BY: JARED ATKINSON
TEMPLATE BY: ANGE ALBERTINI

```
000: 33 C0 8E D0 BC 00 7C 8E C0 8E D8 BE 00 7C BF 00
010: 06 B9 00 02 FC F3 A4 50 68 1C 06 CB FB B9 04 00
020: BD BE 07 80 7E 00 00 7C 0B 0F 85 0E 01 83 C5 10
030: E2 F1 CD 18 88 56 00 55 C6 46 11 05 C6 46 10 00
040: B4 41 BB AA 55 CD 13 5D 72 0F 81 FB 55 AA 75 09
050: F7 C1 01 00 74 03 FE 46 10 66 60 80 7E 10 00 74
060: 26 66 68 00 00 00 00 66 FF 76 08 68 00 00 68 00
070: 7C 68 01 00 68 10 00 B4 42 8A 56 00 8B F4 CD 13
080: 9F 83 C4 10 9E EB 14 B8 01 02 BB 00 7C 8A 56 00
090: 8A 76 01 8A 4E 02 8A 6E 03 CD 13 66 61 73 1C FE
0A0: 4E 11 75 0C 80 7E 00 80 0F 84 8A 00 B2 80 EB 84
0B0: 55 32 E4 8A 56 00 CD 13 5D EB 9E 81 3E FE 7D 55
0C0: AA 75 6E FF 76 00 E8 8D 00 75 17 FA B0 D1 E6 64
0D0: E8 83 00 B0 DF E6 60 E8 7C 00 B0 FF E6 64 E8 75
0E0: 00 FB B8 00 BB CD 1A 66 23 C0 75 3B 66 81 FB 54
0F0: 43 50 41 75 32 81 F9 02 01 72 2C 66 68 07 BB 00
100: 00 66 68 00 02 00 00 66 68 08 00 00 00 66 53 66
110: 53 66 55 66 68 00 00 00 00 66 68 00 7C 00 00 66
120: 61 68 00 00 07 CD 1A 5A 32 F6 EA 00 7C 00 00 CD
130: 18 A0 B7 07 EB 08 A0 B6 07 EB 03 A0 B5 07 32 E4
140: 05 00 07 8B F0 AC 3C 00 74 09 BB 07 00 B4 0E CD
150: 10 EB F2 F4 EB FD 2B C9 E4 64 EB 00 24 02 E0 F8
160: 24 02 C3 49 6E 76 61 6C 69 64 20 70 61 72 74 69
170: 74 69 6F 6E 20 74 61 62 6C 65 00 45 72 72 6F 72
180: 20 6C 6F 61 64 69 6E 67 20 6F 70 65 72 61 74 69
190: 6E 67 20 73 79 73 74 65 6D 00 4D 69 73 73 69 6E
1A0: 67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74
1B0: 65 6D 00 00 00 00 63 7B 9A 82 D4 BA 7D 00 00 00 20
1C0: 21 00 07 FE FF FF 00 08 00 00 00 90 36 06 80 FE
1D0: FF FF 07 FE FF FF 00 A0 36 06 00 60 09 00 00 00
1E0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
1F0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA
```

PARTITION TYPES

0x00 - EMPTY	0x83 - LINUX
0x01 - FAT12	0x84 - HIBERNATION
0x04 - FAT16	0x85 - LINUX_EXTENDED
0x05 - MS_EXTENDED	0x86 - NTFS_VOLUME_SET
0x06 - FAT16	0x87 - NTFS_VOLUME_SET_1
0x07 - NTFS	0xa0 - HIBERNATION_1
0x0b - FAT32	0xa1 - HIBERNATION_2
0x0c - FAT32	0xa5 - FREEBSD
0x0e - FAT16	0xa6 - OPENBSD
0x0f - MS_EXTENDED	0xa8 - MACOSX
0x11 - HIDDEN_FAT12	0xa9 - NETBSD
0x14 - HIDDEN_FAT16	0xab - MAC_OSX_BOOT
0x16 - HIDDEN_FAT16	0xb7 - BSDI
0x1b - HIDDEN_FAT32	0xb8 - BSDI_SWAP
0x1c - HIDDEN_FAT32	0xee - EFI_GPT_DISK
0x1e - HIDDEN_FAT16	0xef - EFI_SYSTEM_PARTITION
0x42 - MS_MBR_DYNAMIC	0xfb - VMWARE_FILE_SYSTEM
0x82 - SOLARIS_X86	0xfc - VMWARE_SWAP
0x82 - LINUX_SWAP	

BOOT CODE

FIELDS	VALUES
jump to boot program	
disk parameters	
boot program code	
disk signature	82D4BA7D

CHS ADDRESSING

```
00100000 00100001 00000000
--------------------->-----
00100000 100001 0000000000
```

Head - 1st byte
Sector - 2nd byte (0-5 bits)
Cylinder - 2nd byte (6-7 bits) 3rd byte

PARTITION TABLE

FIELDS	VALUES
status	0x00 - Non-Bootable
starting head	0x20
starting sector	0x21
starting cylinder	0x00
partition type	0x07 - NTFS
ending head	0xFE
ending sector	0x3F
ending cylinder	0x3FF
relative start sector	0x800
total sectors	0x6369000
status	0x80 - Bootable
starting head	0xFE
starting sector	0x3F
starting cylinder	0x3FF
partition type	0x07 - NTFS
ending head	0xFE
ending sector	0x3F
ending cylinder	0x3FF
relative start sector	0x636A000
total sectors	0x96000
partition type	0x00 - EMPTY
partition type	0x00 - EMPTY

END OF MBR

marker	0x55AA

Fun details

- No distinction between code and data in bootloader code
- 16-bit limitations mean that segmentation addressing must frequently be used
  - `segment:offset` → (`segment` * `16`) + `offset`
  - Commonly, a code segment and a data segment
- Programmer must be very aware of relative addressing

How its actually done

- Because of the 446-byte limitation, actual bootloaders do not usually complete the entire boot process in the boot sector.
- Other approaches are:
  - Use the second+ sectors for more bootloader code, and to load those into memory using first sector code
  - Relocate initial bootloader code, and load a secondary bootloader to `0x7c0:0x0` (good for multibooting)

How its actually done

- Because of the 446-byte limitation, actual bootloaders do not usually complete the entire boot process in the boot sector.
- Other approaches are:
  - Use the second+ sectors for more bootloader code, and to load those into memory using first sector code
  - Relocate initial bootloader code, and load a secondary bootloader to `0x7c0:0x0` (good for multibooting)
  - Give up and blow up your computer (recommended)

```
        org 0x0
        bits 16
        jmp 0x7c0:start          ; set (cs,ip) = (0x7c0, start)

        ;; prepare to copy sector at 0x7c0:0x0 to 0x060:0x0
start:

        ;; set data segment to point to code segment 0x7c0:0x0
        mov bx, 0x7c0
        mov ds, bx


        ;;  store boot drive number for later use in BIOS calls
        mov [ds:511], dl

        ;; clear interrupt and direction flags
        cli
        cld
```

```
copy_sector:
        ;; finish copying if we've copied 512 bytes
        cmp bx, 512
        je relocate

        ;; copy one byte from ds sector to es sector
        mov ah, [ds:bx]
        mov [es:bx], ah

        ;; move bx to point to the next byte, and loop
        inc bx
        jmp copy_sector
```

```
        ;; set code segment and data segment to point
        ;; to freshly copied segment 0x060:0x0
relocate:

        ;; set the data segment
        mov bx, 0x060
        mov ds, bx

        ;; set up (small) stack
        mov sp, 446

        ;; (implicitly) set code segment through this jump
        jmp 0x060:find_active_partition
```

```
        ;; Wait for user to press a key, then reboot
reboot_on_keypress:
        xor ah, ah
        int 0x16
        int 0x19
        hlt
```


QEMU
Machine View
Welcome to RunicOS!
Disk read success...

Credits

- http://www.invoke-ir.com/2015/05/ontheforensictrail-part2.html
- https://wiki.osdev.org/Boot_Sequence
- https://www.teachoo.com/10705/3113/Electric-Generator/category/Concepts/