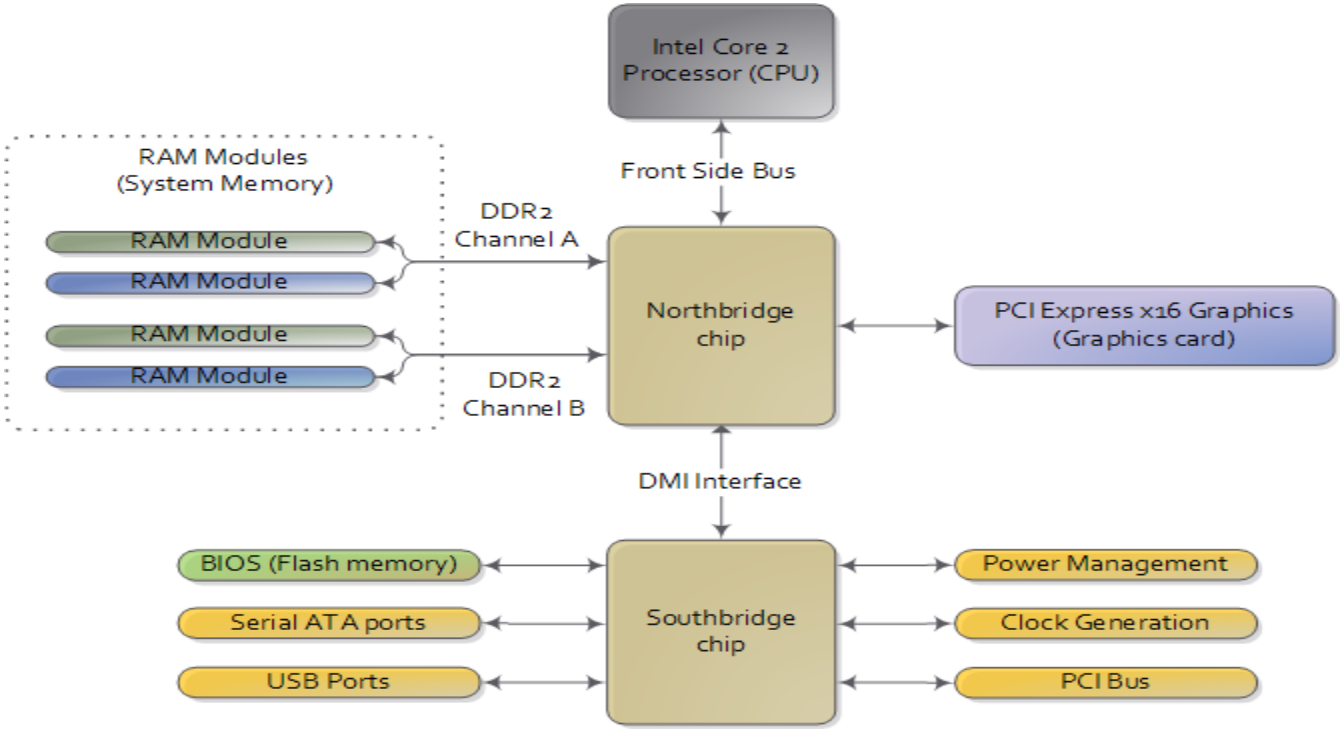
ОС

Загрузчик

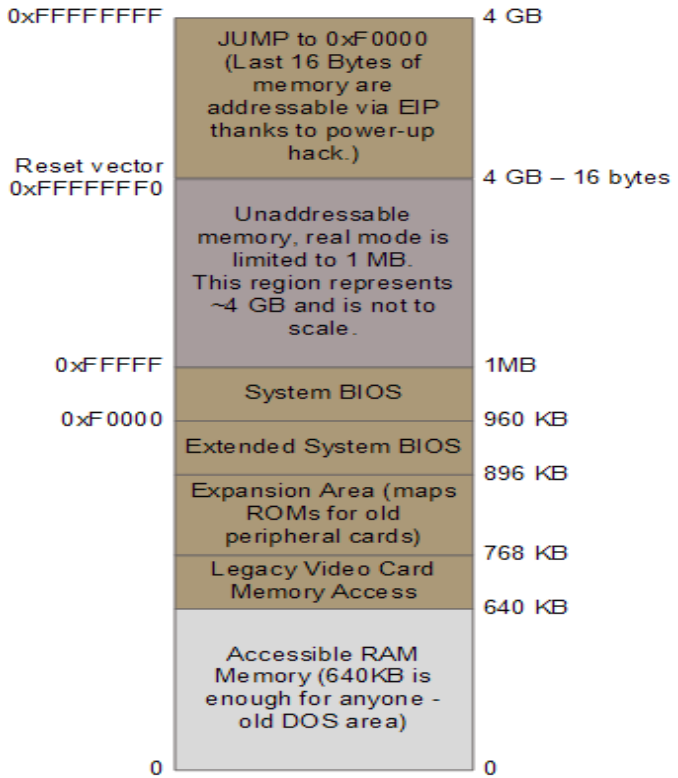
Motherboard



[1] http://duartes.org/gustavo/blog/post/motherboard-chipsets-memory-map

Memory map

- В Real Mode адресация 20 битная => 1MB - верхняя граница допустимой памяти
- Используется сегментная адресация - сегментный регистр и смещение (например, для пары CS:IP, Addr = CS << 4 + IP)
- Загрузчик работает в нижних адресах памяти (0-640KB)



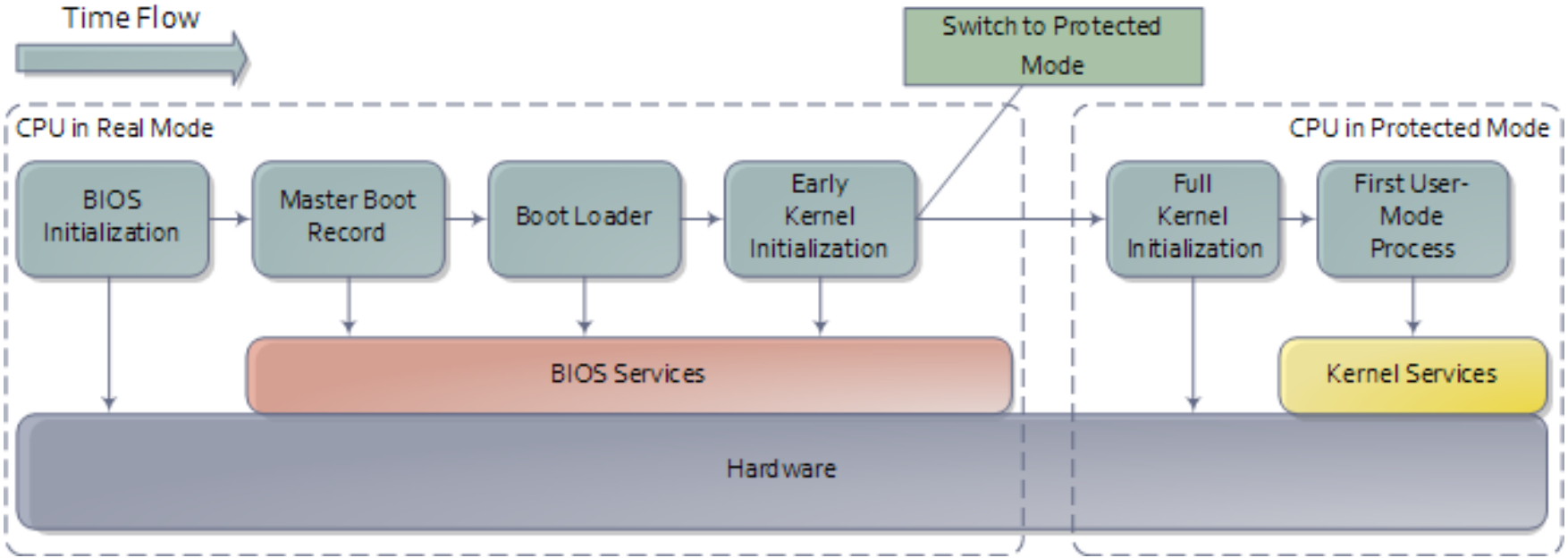
[2] http://duartes.org/gustavo/blog/post/how-computers-boot-up

BIOS

- BIOS - Basic Input/Output System:
  - унифицированный доступ к низкоуровневым сервисам
  - скрывает аппаратные различия (на сколько это возможно)
  - недоступен “почти” в protected mode

- Примеры сервисов (see http://wiki.osdev.org/BIOS):
  - видеоадаптер (int 0x10)
  - система хранения (диски, дискеты, usb) (int 0x13)
  - размер основной памяти (int 0x15)
  - клавиатура (int 0x16)

Загрузчик

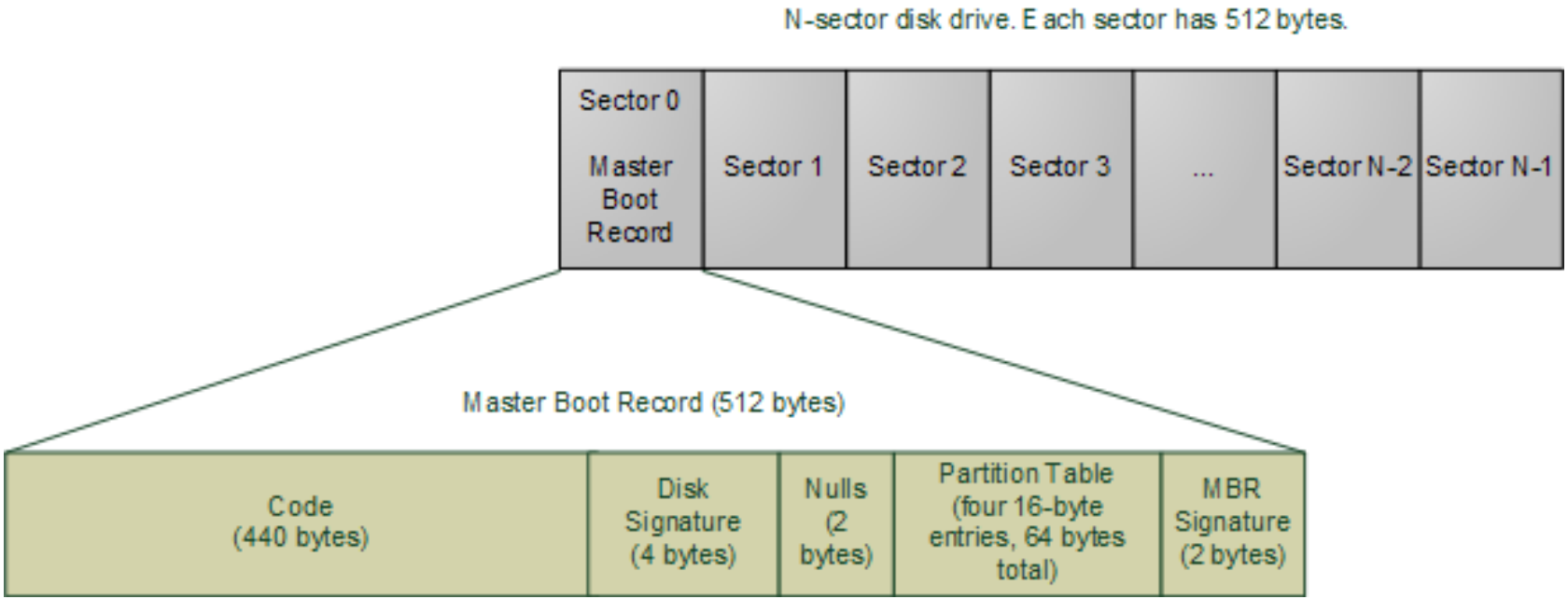


[3] http://duartes.org/gustavo/blog/post/how-computers-boot-up

Начальная загрузка

1. [Power On =>] Reset CPU
2. Немного специфичной магии (зависит от версии CPU)
3. Передача управления по адресу 0xffff0 [cs = 0xf000, ip = 0xfff0] - BIOS Sturtup Code
4. Power-on self-test (POST)
5. Опредление устройств с собственными BIOS и их инициализация
6. Memory Test
7. Настройка параметров устройств
8. Выбор загрузочного устройства
9. Передача управления загрузчику (Jump at 0x7c00)

Master Boot Record



[4] http://duartes.org/gustavo/blog/post/how-computers-boot-up

Bootloader stages

- MBR содержит первичный загрузчик (first stage bootloader):
  - ограничен в размере (440 байт)
  - загружает основной загрузчик (например, NTLDR) и передает ему управление

- Основной загрузчик:
  - имеет доступ к файловой системе
  - по-факту является миниатюрной ОС
  - загружает ядро ОС (например, c:\Windows\System32\ntoskrnl.exe или /boot/vmlinuz-3.11.0-12-generic)

Multiboot

- Открытая спецификация описывающая интерфейс взаимодействия ОС и загрузчика:
  - позволяет использовать один загрузчик для всех совместимых ОС (see http://www.gnu.org/software/grub/manual/multiboot/multiboot.html)
  - поддерживается различными ОС (Linux, OpenSolaris, GNU Hurd и др)
  - GNU GRUB - эталонная реализация

- От бинарного файла требуется наличие заголовка multiboot
- Определяет состояние CPU в момент передачи управления ядру

(U)EFI

- (Unified) Extensible Firmware Interface
  - определена для разных платформ (Itanium, x86 32/64, ARM)
  - поддерживается современными ОС (Windows, Mac OS X, FreeBSD, Linux)
  - поддерживается загрузчиками GRUB и LILO
  - поддерживается эмуляторами VirtualBox и QEMU
- Поддерживает универсальные (независиыме от CPU) программы (Efi Byte Code)
- Модульный дизайн
- 32 или 64 битный режимы

Практика

```
        .code16
        .text
        .globl _start
_start:
        cli                             /* 1. disable interrupts */
        movw  %cs,          %ax         /* 2. setup data segment */
        movw  %ax,          %ds
        addw  %0x0220,      %ax         /* 3. setup stack segment */
        movw  %ax,          %ss
        movw  %0x0100,      %sp         /* 4. setup stack pointer */
        sti                             /* 5. enable interrupts */
        . . .                           /* 6. loading here */
        hlt
        . . .                           /* some other stuff */
        . = _start + 0x0200 - 2
        .short0xAA55
```

Как это собрать?

- Указать ld выходной формат - binary:

```
--oformat binary
```

- Указать ld ld-скрипт:

```
-T script.ld
```

- ld-скрипт “размечает” бинарный файл:

```
SECTIONS
{
    . = 0x7C00;
    .text : { *(.text) }
}
```

Как проверить?

```
sudo apt-get install qemu

qemu-system-i386 -hda ./boot.bin


sudo dd if=./boot.bin of=/dev/sdb

# and boot from /dev/sdb
```

Q&A