

# Операционные Системы

## Процесс загрузки

April 28, 2017

# Первая инструкция

- ▶ Откуда берется первая инструкция?

# Первая инструкция

- ▶ Откуда берется первая инструкция?
  - ▶ x86 обращается по адресу *0xFFFFFFF0*;

# Первая инструкция

- ▶ Откуда берется первая инструкция?
  - ▶ x86 обращается по адресу *0xFFFFFFF0*;
  - ▶ отвечает ему *материнская карта*.

# Первая инструкция

- ▶ Откуда берется первая инструкция?
  - ▶ x86 обращается по адресу *0xFFFFFFF0*;
  - ▶ отвечает ему *материнская карта*.
- ▶ Какой код материнская карта отдает процессору?
  - ▶ BIOS (Basic Input/Output System) - наследство *IBM PC*;
  - ▶ UEFI (Unified Extensible Firmware Interface).

# BIOS

- ▶ POST (Power-On Self-Test)
  - ▶ проверяет, что все на месте и "работает";
  - ▶ может выполнять начальную инициализацию устройств;

# BIOS

- ▶ POST (Power-On Self-Test)
  - ▶ проверяет, что все на месте и "работает";
  - ▶ может выполнять начальную инициализацию устройств;
  - ▶ ищет загрузочное устройство (диск в ОС).

# Загрузочное устройство

- ▶ BIOS ищет диск, с которого можно прочитать первые 512 байт
  - ▶ а. к. а. загрузочный сектор;



# Загрузочное устройство

- ▶ BIOS ищет диск, с которого можно прочитать первые 512 байт
  - ▶ а. к. а. загрузочный сектор;
  - ▶ последние 2 байта сектора должны хранить числа  $0x55$  и  $0xAA$ ;

# Загрузочное устройство

- ▶ BIOS ищет диск, с которого можно прочитать первые 512 байт
  - ▶ а. к. а. загрузочный сектор;
  - ▶ последние 2 байта сектора должны хранить числа *0x55* и *0xAA*;
  - ▶ сектор загружается в память по физическому адресу *0x7c00*.

# Загрузочное устройство

- ▶ BIOS ищет диск, с которого можно прочитать первые 512 байт
  - ▶ а. к. а. загрузочный сектор;
  - ▶ последние 2 байта сектора должны хранить числа *0x55* и *0xAA*;
  - ▶ сектор загружается в память по физическому адресу *0x7c00*.
- ▶ BIOS передает управление по физическому адресу *0x7c00*
  - ▶ мы добрались до места, где мы можем на что-то повлиять.

# Окружение

- ▶ Что нам известно о состоянии системы?

# Окружение

- ▶ Что нам известно о состоянии системы?
  - ▶ наш код начинается по физическому адресу *0x7c00*;

# Окружение

- ▶ Что нам известно о состоянии системы?
  - ▶ наш код начинается по физическому адресу `0x7c00`;
  - ▶ устройства как-то инициализированы и прерывания отключены;

# Окружение

- ▶ Что нам известно о состоянии системы?
  - ▶ наш код начинается по физическому адресу *0x7c00*;
  - ▶ устройства как-то инициализированы и прерывания отключены;
  - ▶ процессор работает в *Real Mode*.

# Real Mode

- ▶ Логический адрес состоит из двух частей:
  - ▶ 16-битного сегмента (*SEG*) и 16-битного смещения (*OFF*);



# Real Mode

- ▶ Логический адрес состоит из двух частей:
  - ▶ 16-битного сегмента ( $SEG$ ) и 16-битного смещения ( $OFF$ );
  - ▶ физический адрес получается по формуле  $(SEG * 16 + OFF) \bmod 2^{20}$ .

# Real Mode

- ▶ Логический адрес состоит из двух частей:
  - ▶ 16-битного сегмента ( $SEG$ ) и 16-битного смещения ( $OFF$ );
  - ▶ физический адрес получается по формуле  $(SEG * 16 + OFF) \bmod 2^{20}$ .
- ▶ Сегмент хранится в одном из специальных регистров:
  - ▶  $CS, DS, SS, ES, FS, GS$ .

# Real Mode

- ▶ Регистры общего назначения 16-битные:
  - ▶ *SP* - указатель стека;
  - ▶ *BP* - указатель "базы";
  - ▶ *AX, BX, CX, DX, SI, DI*.

# Hello, World!

```
.code16
.text
.global start

start:
    ljmp 0x0, $real_start

real_start:
    movw $0, %ax
    movw %ax, %ds
    movw %ax, %ss

    movw $0x7c00, %sp
    addw $0x0400, %sp

    ...

loop:  jmp loop
```

# Hello, World!

```
movw $0xB800 , %ax
movw %ax , %es
movw $data , %si
movw $0 , %di
movw size , %cx
call memcpy
```

...

data:

```
.asciz "H\017e\017l\017l\017o
      ↪ \017!\017"
```

size:

```
.short . - data
```

# Hello, World!

memcpy:

```
    cmpw $0, %cx  
    jz  out
```

next:

```
    movb (%si), %ah  
    movb %ah, %es:(%di)  
    incw %si  
    incw %di  
    decw %cx  
    jnz next
```

out:

```
    ret
```

# Начальный загрузчик

- ▶ Как много кода можно поместить в первые 510 байт?
  - ▶ вряд ли туда поместится целая современная ОС;

# Начальный загрузчик

- ▶ Как много кода можно поместить в первые 510 байт?
  - ▶ вряд ли туда поместится целая современная ОС;
  - ▶ задача этого кода прочитать с диска код, не поместившийся в первые 510 байт.



# Начальный загрузчик

- ▶ Как много кода можно поместить в первые 510 байт?
  - ▶ вряд ли туда поместится целая современная ОС;
  - ▶ задача этого кода прочитать с диска код, не поместившийся в первые 510 байт.
- ▶ Оставшийся код может быть кодом ОС,
  - ▶ а может быть кодом (вторичного) загрузчика;

# Начальный загрузчик

- ▶ Как много кода можно поместить в первые 510 байт?
  - ▶ вряд ли туда поместится целая современная ОС;
  - ▶ задача этого кода прочитать с диска код, не поместившийся в первые 510 байт.
- ▶ Оставшийся код может быть кодом ОС,
  - ▶ а может быть кодом (вторичного) загрузчика;
  - ▶ например, GRUB.