Санкт-Петербургский политехнический университет Петра Великого
Кафедра компьютерных систем и программных технологий

Отчет по дисциплине
«Проектирование ОС и их компонентов»

Разработка своего загрузчика (Asm/C/C++)
(разработка под Linux)

Работу выполнил студент группы №: 13541/3 Чеботарёв М. М.
Работу принял преподаватель: _____ Душутина Е. В.

Санкт-Петербург
2017 г.

1. Используемая система и версия ядра

a) Linux

```
michael@michael-LIFEBOOK-AH531:~$ lsb_release  -a
No LSB modules are available.
Distributor ID:   Ubuntu
Description:      Ubuntu 16.04.1 LTS
Release:          16.04
Codename:         xenial


michael@michael-LIFEBOOK-AH531:~$ cat /proc/version
Linux version 4.4.0-38-generic (buildd@lgw01-58) (gcc version 5.4.0 20160609
(Ubuntu 5.4.0-6ubuntu1~16.04.2) ) #57-Ubuntu SMP Tue Sep 6 15:42:33 UTC 2016
```

2. Загрузка с floppy disk диска

2.1. Создание виртуального Floppy-диска

Для выполнения данной задачи можно воспользоваться двумя методами:

- создать образ с помощью внешней программы WinImage;
- создать образ средствами VMWare;

Первым шагом данного этапа является создание пустого образа floppy диска. Сразу пойдем 2ым путем: в свойства виртуальной машины создадим новый Floppy-диск и укажем его в качестве монтируемого (рис.1).

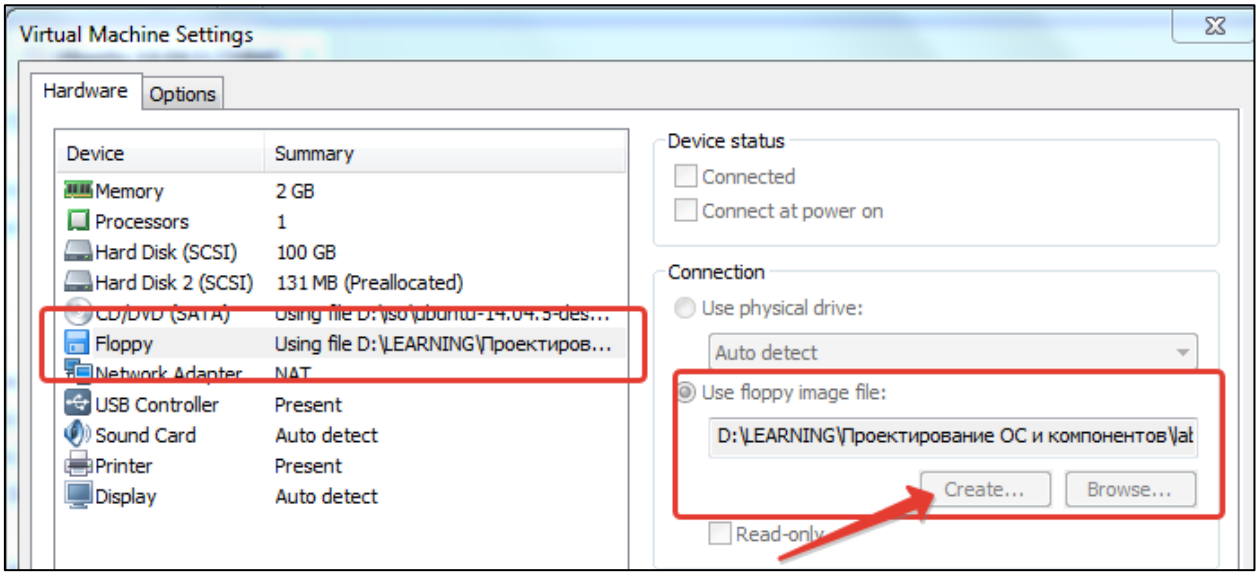

Рисунок 1. Создание/добавление нового floppy-диска

2.2. Задание приоритетов устройств при загрузке

ПО умолчанию загрузка с гибких носителей имеет бОльший приоритет. В целом проверить это возможно следующим способом: в меню запуска выбрать пункт «Включить

питание из BIOS» (рис.2), и после загрузки BIOS выбрать раздел Boot, где уже можно задать приоритеты.

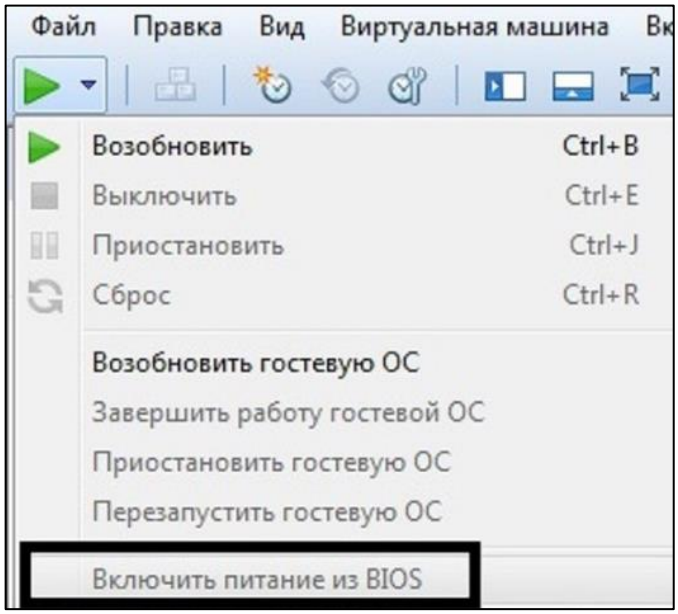

Рисунок 2. Загрузка BIOS

После основных настроек можем приступать к написанию кода загрузчика.

2.3. Описание процесса загрузки

Рассмотрим процесс загрузки более подробно. При подаче на устройства питания начинает выполняться BIOS (Basic Input / Output System), которая представляет собой мини-ОС, встроенную в систему. Он выполняет некоторые аппаратные тесты, далее начинается загрузка операционной системы с любого носителя. Это может быть либо жесткий диск, либо floppy диск, flash диск.

BIOS пытается найти Master Boot Record (MBR), раздел размером 512 байт в начале диска, загружает его свою область памяти по физическому адресу 0x7C00, и передаёт управление по физическому адресу 0x7C00 (то есть сектору MBR), предварительно записав в регистр DL номер диска, с которого этот сектор считан. MBR - программа, которая загружает основное ядро операционной системы. Загрузчик должен оканчиваться сигнатурой 55AAh.

Для простоты первоначально напишем код загрузчика, который выводит строку “Hello World!” на экран. Используемый язык – nasm.

Листинг №1. boot.asm

```
[BITS 16]           ;16-битный код
[ORG 0x7C00]        ;указатель на адрес, где будет находиться код,
                    ;после загрузки системы
MOV SI, HelloString ;сохранение указателя на строку в SI

CALL PrintString
```

```
 JMP $

PrintCharacter:     ;процедура для вывода символа на экран
                    ;необходимое ASCII значение должно быть в регистре AL
MOV AH, 0x0E        ;говорит BIOS что выводим один символ на экран
MOV BH, 0x00        ;нет страниц
MOV BL, 0x07        ;выбор цвета

INT 0x10            ;вызов прерывания 0x10
RET                 ;выход из процедуры

PrintString:        ;процедура для вывода строки на экран
                    ;указатель на строку в регистре SI

next_character:     ;
MOV AL, [SI]        ;получение и сохранения байта строки в регистре AL
INC SI              ;Инкремент указателя на строку
OR AL, AL           ;проверка, что в регистре AL -  0  (конец строки)
JZ exit_function    ;завершение процедуры
CALL PrintCharacter ;вывод очередного символа
JMP next_character  ;вывести следующий символ

exit_function:      ;
RET                 ;выход из процедуры

;Data
HelloString db 'Hello World!', 0       ;строка HelloWorld!, оканчивается 0

TIMES 510 - ($ - $$) db 0              ;заполняет оставшееся место на секторе 0
DW 0xAA55                              ;добавление спец. сигнатуры в конце загрузчика
```

Обратим внимание, что для вывода строки на экран вызывается прерывание 0x10. Перед вызовом прерывания необходимо установить следующие значения в регистры

AH - 0x0E ; выводим один символ на экран
BH - 0x00 ; нет страниц
BL - 0x07 ; выбор цвета.

2.5. Установка средств разработки

2.5.1. Ассемблер NASM

Доступно 2 способа:

- Установка из архива
- Установка пакета командой apt-get install nasm

В данном случае вторая команда не прошла, поэтому **пришлось вручную установить** компилятор. Необходимо выполнить следующие действия:

1. Скачать архив ***.tar.gz** из репозитория The netwide assembler(NASM);
2. Разархивировать командой **tar –xvf *.tar.gz**
3. Войти в созданную папку командой **cd**;
4. выполнить команду **./configure**, которая создаст Makefile`ы;

5. выполнить команду **make**;
6. выполнить команду **make install**.

2.5.2. QEMU

QEMU — система эмуляции (и виртуализации) компьютера (вычислительной системы с процессором, памятью и периферийными устройствами), поддерживающего различные архитектуры. Для установки следует выполнить команду:

sudo apt-get install qemu-system

2.6. Компиляция и тестирование V1.0

Компилируем файл следующей командой

nasm boot.asm boot.bin

После необходимо поместить созданный bin файл на floppy диск. Выполним команду dd со следующими параметрами:

- if — чтение файла boot.bin
- of — запись в файл /dev/fd0 (используемый floppy disk)
- bs — размер блока 512.

dd if=boot.bin bs=512 of=/dev/fd0

```
osboxes@osboxes:~/Desktop/SysteProgramming$ ls -l /dev/fd*
lrwxrwxrwx 1 root root     13 May 29 11:41 /dev/fd -> /proc/self/fd
brw-rw---- 1 root floppy 2, 0 May 29 11:41 /dev/fd0
osboxes@osboxes:~/Desktop/SysteProgramming$ ls -l
total 12
-rwxrw-rw- 1 osboxes osboxes 1711 May 29 11:33 boot.asm
-rw-rw-r-- 1 osboxes osboxes  512 May 29 11:33 boot.bin
osboxes@osboxes:~/Desktop/SysteProgramming$ sudo dd if=boot.bin of=/dev/fd0 bs=512
count=1
[sudo] password for osboxes:
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.0164586 s, 31.1 kB/s
```

Перезагружаем компьютер и должны увидеть надпись Hello World, но я ее не увидел, загрузка с Floppy прошла, однако ничего не отобразилось. Поэтому предпринята другая попытка: загрузить загрузчик, и от него передать управление в другую программу, которая будет эмулировать ядро и выведет желаемую надпись.

2.7. Компиляция и тестирование V2.0

На floppy диск будет записано 2 сектора, каждый по 512 байт. Первый сектор – загрузчик, должен передать управление второму сектору. Второй сектор, назовем его kernel. Для простоты реализации второй сектор просто выводит строку на экран так, как это было реализовано в предыдущем пункте.

Листинг №2. bootV2.asm

```
[BITS 16]
[ORG 0x7C00]
MOV DL, 0x0         ;drive 0 = floppy
MOV DH, 0x0         ;head (0=base)
MOV CH, 0x0         ;track/cylinder
MOV CL, 0x02        ;номер сектора (нумерация начинается с 1)
MOV BX, 0x1000      ;адрес RAM для загрузки ядра
MOV ES, BX          ;старший адрес RAM 0x1000
MOV BX, 0x0         ;младший адрес RAM 0x0
ReadFloppy:
MOV AH, 0x02        ;чтение сектора диска
MOV AL, 0x01        ;количество секторов для чтения
INT 0x13

;указатель на адрес RAM
MOV AX, 0x1000
MOV DS, AX
MOV ES, AX
MOV FS, AX
MOV GS, AX
MOV SS, AX
JMP 0x1000:0x0
TIMES 510 - ($ - $$) db 0 ;fill resting bytes with zero
DW 0xAA55           ;конец загрузчика
```

Для вызова следующей стадии загрузчика используется прерывание 0x13 с параметрами:

DL = 0x0 ; диск 0 = floppy
DH = 0x0 ; head (0=base)
CH = 0x0 ; track/cylinder
CL = 0x02 ; номер сектора (нумерация начинается с 1)
BX = 0x0 ; младший адрес RAM
ES = 0x1000 ; старший адрес RAM 0x1000
AH=0x02 ; чтения сектора с диска
AL=0x01 ; количество секторов для чтения.

Примечание: указали, что необходимо прочитать один сектор (под номером 2) из floppy диска и записать прочитанное в RAM 0x1000:0x0. После чего осуществить переход по адресу 0x1000:0x0.
Теперь необходимо написать вторую ступень загрузчика, осуществляющий вывод на экран строки (код соответствует коду загрузчика предыдущего этапа).

Листинг №3. kernel.asm

```
MOV SI, HelloString       ;сохранение указателя на строку в SI

CALL PrintString
 JMP $

PrintCharacter:           ; процедура для вывода символа на экран
                          ;необходимое ASCII значение должно быть в регистре AL
MOV AH, 0x0E              ;говорит BIOS что выводим один символ на экран
```

```
MOV BH, 0x00              ;нет страниц
MOV BL, 0x07              ;выбор цвета

INT 0x10                  ;вызов прерывания 0x10
RET                       ;выход из процедуры

PrintString:              ; процедура для вывода строки на экран
                          ;указатель на строку в регистре SI

next_character:           ;
MOV AL, [SI]              ;получение и сохранения байта строки в регистре AL
INC SI                    ;Инкремент указателя на строку
OR AL, AL                 ;проверка, что в регистре AL -  0  (конец строки)
JZ exit_function          ;завершение процедуры
CALL PrintCharacter       ;вывод очередного символа
JMP next_character        ;вывести следующий символ

exit_function:            ;
RET                       ;выход из процедуры

;Data
HelloString db 'Hello World!', 0      ;строка HelloWorld!, оканчивается 0
TIMES 510 - ($ - $$) db 0             ;заполняет оставшееся место на секторе 0
```

Файл kernel.asm аналогичен предыдущему примеру вывода строки на экран.
Компилируем созданные файлы командой

nasm boot.asm –o boot.bin

nasm kernel.asm –o kernel.bin

После необходимо поместить созданные bin файлы на floppy диск. Файл boot.bin поместим в первый сегмент диска, kernel.asm поместим во второй сегмент диска. Для этого выполним команду dd со следующими параметрами:

- if — чтение файла boot.bin
- of — запись в файл /dev/fd0 (используемый floppy disk)
- bs — размер блока 512
- count – 1, копируем 1 блок
- seek – 1, перед копированием пропускаем 1 блок от начала в выходном файле.

dd if=boot.bin bs=512 of=/dev/fd0 count=1
dd if=kernel.bin bs=512 of=/dev/fd0 count=1 seek=1

Результат исполнения:

```
osboxes@osboxes:$ sudo dd if=boot.bin of=/dev/fd0 bs=512 count=1
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.00110442 s, 464 kB/s
osboxes@osboxes:$ sudo dd if=kernel.bin of=/dev/fd0 bs=512 count=1 seek=1
1+0 records in
1+0 records out
512 bytes (512 B) copied, 0.00111469 s, 459 kB/s
osboxes@osboxes:$ sudo  reboot
```

Перезагружаем компьютер и видим надпись “Hello World!” (рис 3).

Рисунок 3. Результат работы загрузчика

BIOS удалось обнаружить на floppy диске созданный загрузчик (boot.bin). Загрузчик в свою очередь передал управление второй ступени (kernel.bin), которая вывела на экран строку “Hello World!”.

2.8. Загрузка с жесткого диска

Для эмуляции жесткого диска воспользуемся программой QEMU.

QEMU — свободная программа с открытым исходным кодом для эмуляции аппаратного обеспечения различных платформ. Включает в себя эмуляцию процессоров Intel x86 и устройств ввода-вывода.

QEMU предоставляет специальную команду для создания жесткого диска, которая называется qemu-img. Эта утилита создает образы различных форматов. Для quemu-img необходимо указать операцию (create для создания нового образа диска), формат (qcow для форматирования образа qemu), размер и имя образа диска. Следующий пример создает образ диска на 128 Мб.

```
osboxes@osboxes:$ qemu-img create -f qcow disk.img 128M
Formatting 'disk2.img', fmt=qcow size=134217728 encryption=off
```

После командой dd, как было рассмотрено выше, поместим *.bin файлы на образ диска.

dd if=boot.bin bs=512 of= disk.img count=1
dd if=kernel.bin bs=512 of= disk.img count=1 seek=1

Запустим созданный образ диска, используя QEMU эмулятор.

sudo qemu-system-i386 -hda disk.img

После чего открывается окно, которое эмулирует работу созданной системы (рис. 4-5)

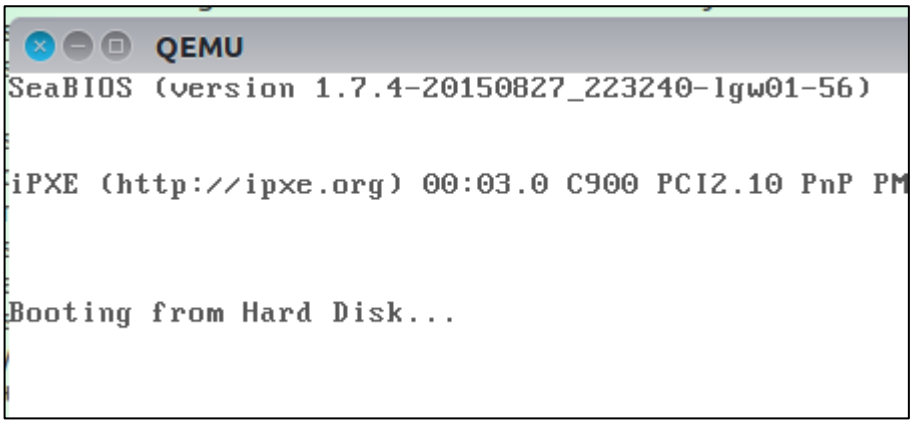

Рисунок 4. Пример не удачной загрузки

Рисунок 5. Пример удачной загрузки

Мы должны увидеть, что загрузчик с жесткого диска успешно вывел на экран строку “Hello World!”.

Выводы:

Мы рассмотрели, что такое загрузчик, как работает BIOS, и как компоненты системы взаимодействуют при загрузке системы. Практическая часть дала понимание о том, как можно разработать свой собственный, простой загрузчик. Описано, что происходит после включения компьютера и как система загружается. В качестве практического примера, рассмотрено как можно написать свой собственный загрузчик, который фактически является отправной точкой при загрузки системы, а помимо этого представлен пример загрузки своей «операционной системы» из созданного загрузчика.

Конечно, это лишь малая часть по сравнению с огромной темой низкоуровневого программирования.

ИСТОЧНИКИ

1. Compiling an assembly program with Nasm

http://ccm.net/faq/1559-compiling-an-assembly-program-with-nasm

2. How to develop your own Boot Loader

https://www.codeproject.com/Articles/36907/How-to-develop-your-own-Boot-Loader

3. QEMU/QEMU Hello World: Installing QEMU and getting it up and running

https://en.wikibooks.org/wiki/QEMU/QEMU_Hello_World:_Installing_QEMU_and_getting_it_up_and_running

4. Twenty one pilots: Heathens (from Suicide Squad: The Album)

https://www.youtube.com/watch?v=UprcpdwuwCg&index=25&list=RD9sg-A-eS6Ig

5. QEMU

http://xgu.ru/wiki/QEMU ***-***