

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

МОСКОВСКИЙ АВИАЦИОННЫЙ ИНСТИТУТ (НАУЧНО-ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ)

ЖУРНАЛ
ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Наименование практики *вычислительная*

Студенты:

Ногаев Дамир Таймуразович

Арешин Станислав Олегович

Литвина Анастасия Алексеевна

Факультет № 8 курс 2 группа 6

Практика с 27.06.19 по 12.07.19

ИНСТРУКЦИЯ

о заполнении журнала по производственной практике

Журнал по производственной практике студентов имеет единую форму для всех видов практик.

Задание в журнал вписывается руководителем практики от института в первые три – пять дней пребывания студентов на практике в соответствии с тематикой, утверждённой на кафедре до начала практики. Журнал по производственной практике является основным документом для текущего и итогового контроля выполнения заданий, требований инструкции и программы практики.

Табель прохождения практики, задание, а также технический отчёт выполняются каждым студентом самостоятельно.

Журнал заполняется студентом непрерывно в процессе прохождения всей практики и регулярно представляется для просмотра руководителям практики. Все их замечания подлежат немедленному выполнению.

В разделе «Табель прохождения практики» ежедневно должно быть указано, на каких рабочих местах и в качестве кого работал студент. Эти записи проверяются и заверяются цеховыми руководителями практики, в том числе мастерами и бригадирами. График прохождения практики заполняется в соответствии с графиком распределения студентов по рабочим местам практики, утверждённым руководителем предприятия.

В разделе «Рационализаторские предложения» должно быть приведено содержание поданных в цехе рационализаторских предложений со всеми необходимыми расчётами и эскизами. Рационализаторские предложения подаются индивидуально и коллективно.

Выполнение студентом задания по общественно-политической практике заносятся в раздел «Общественно-политическая практика». Выполнение работы по оказанию практической помощи предприятию (участие в выполнении спецзаданий, работа сверхурочно и т.п.) заносятся в раздел журнала «Работа в помощь предприятию» с последующим письменным подтверждением записанной работы соответствующими цеховыми руководителями.

Раздел «Технический отчёт по практике» должен быть заполнен особо тщательно. Записи необходимо делать чернилами в сжатой, но вместе с тем чёткой и ясной форме и технически грамотно. Студент обязан ежедневно подробно излагать содержание работы, выполняемой за каждый день. Содержание этого

раздела должно отвечать тем конкретным требованиям, которые предъявляются к техническому отчёту заданием и программой практики. Технический отчёт должен показать умение студента критически оценивать работу данного производственного участка и отразить, в какой степени студент способен применить теоретические знания для решения конкретных производственных задач.

Иллюстративный и другие материалы, использованные студентом в других разделах журнала, в техническом отчёте не должны повторяться, следует ограничиваться лишь ссылкой на него. Участие студентов в производственно-технической конференции, выступление с докладами, рационализаторские предложения и т.п. должны заноситься на свободные страницы журнала.

Примечание. Синьки, кальки и другие дополнения к журналу могут быть сделаны только с разрешения администрации предприятия и должны подшиваться в конце журнала.

Руководители практики от института обязаны следить затем, чтобы каждый цеховой руководитель практики перед уходом студентов из данного цеха в другой цех вписывал в журнал студента отзывы об их работе в цехе.

Текущий контроль работы студентов осуществляется руководителями практики от института и цеховыми руководителями практики заводов. Все замечания студентам руководители делают в письменном виде на страницах журнала, ставя при этом свою подпись и дату проверки.

Результаты защиты технического отчёта заносятся в протокол и одновременно заносятся в ведомость и зачётную книжку студента.

Примечание. Нумерация чистых страниц журнала проставляется каждым студентом в своём журнале до начала практики.

С инструкцией о заполнении журнала ознакомился:

«12» июля 2019г.

Студент Ногаев _____

(подпись)

Студент Арешин _____

(подпись)

Студент Литвина _____

ЗАДАНИЕ

кафедры 806 по вычислительной практике

1. Создать загрузочный образ миниядра MiniX.
1. Изучить механизм прерывания страничной организации памяти.
2. Составить алгоритм.
3. Реализовать управление жестким диском.
4. Тестирование программы.
5. Список используемой литературы.
6. Выводы.

**Руководитель практики
от института**

«12» июля 2019 г.

Подпись

Отзывы цеховых руководителей практики

Работа в помощь предприятию

ПРОТОКОЛ

ЗАЩИТЫ ТЕХНИЧЕСКОГО ОТЧЁТА

по производственной практике

студентами Ногаевым Дамиром Таймуразовичем
Арешиным Станиславом Олеговичем
Литвиной Анастасией Алексеевной

(фамилия, имя и отчество)

Слушали:	Постановили:
Отчёт практиканта	Считать практику выполненной и защищённой на
1. Создать загрузочный образ миниядра MiniX.	Ногаев Д.Т. Оценка _____
	Арешин С.О. Оценка _____
	Литвина А.А. Оценка _____
2. Изучить механизм прерывания страничной организации памяти. Построить модель алгоритма.	Ногаев Д.Т. Оценка _____
	Арешин С.О. Оценка _____
	Литвина А.А. Оценка _____
3. Составить алгоритм.	Ногаев Д.Т. Оценка _____
	Арешин С.О. Оценка _____
	Литвина А.А. Оценка _____
4. Реализовать управление жестким диском.	Ногаев Д.Т. Оценка _____
	Арешин С.О. Оценка _____
	Литвина А.А. Оценка _____
5. Тестирование программы.	Ногаев Д.Т. Оценка _____
	Арешин С.О. Оценка _____
	Литвина А.А. Оценка _____
6. Список используемой литературы.	

7. Выводы.

Общая оценка _____

Руководитель: Семенов А. С. _____
Дата: 12.07.2019

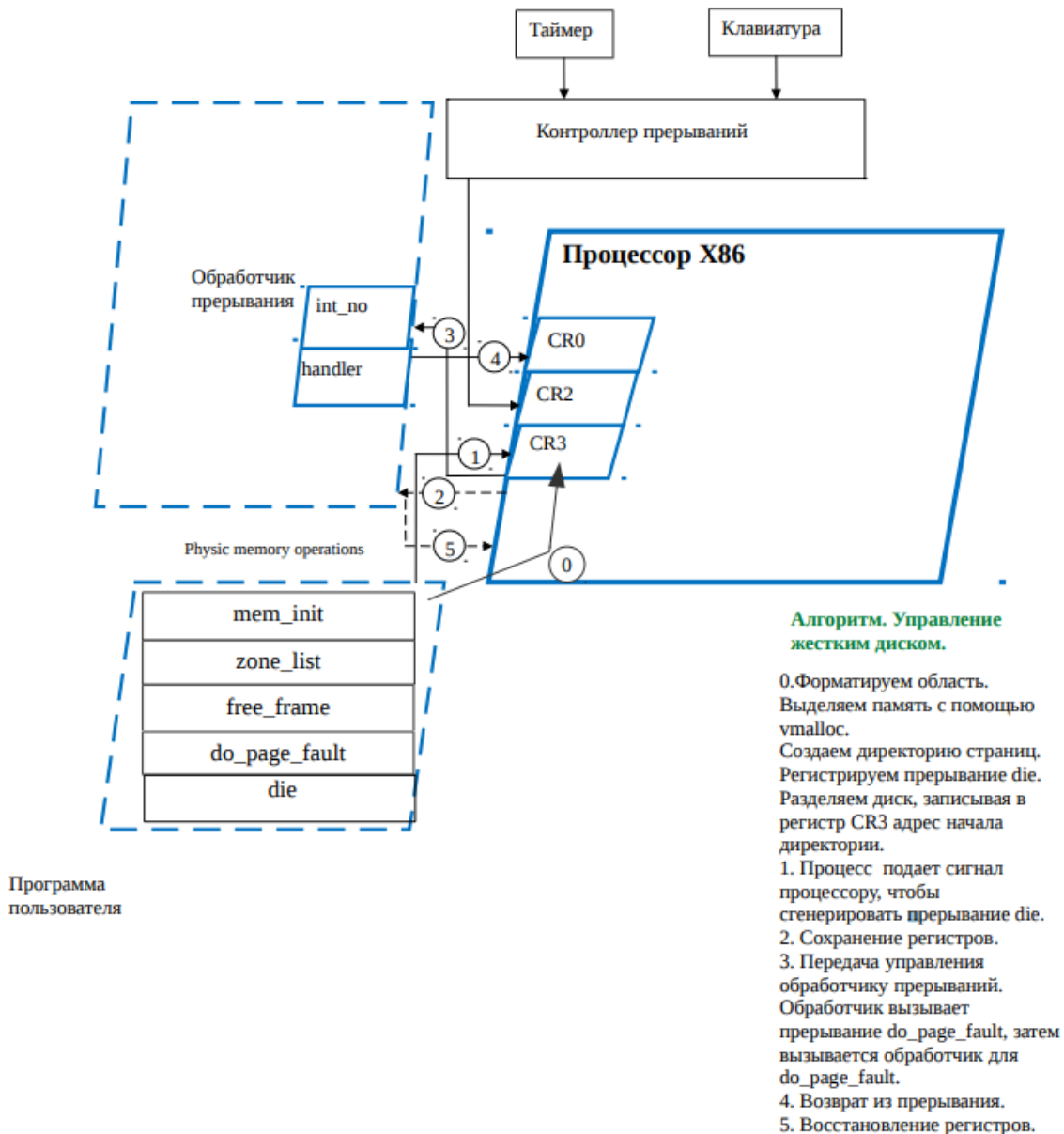
МАТЕРИАЛЫ ПО РАЦИОНАЛИЗАТОРСКИМ ПРЕДЛОЖЕНИЯМ

Скрипт для установки образа на новых версиях ОС (Ubuntu 19.04).

```
#!/bin/bash
echo "Creating empty image file...";
dd if=/dev/zero of=./hdd.img bs=512 count=16065
echo "Attaching file using first found loop device..."
device=`sudo losetup --find --show ./hdd.img`
echo "Creating partition table and bootable partition..."
(echo o; echo w;) | \
sudo fdisk $device
(echo n; echo p; echo 1; echo ; echo ; echo a; echo w;) | \
sudo fdisk $device
echo "Reattaching device and newly created partitions..."
sudo losetup -d $device
device=`sudo losetup --find --show --partscan ./hdd.img`
partition="$device"p1
echo "Formatting partition to ext4 filesystem..."
sudo mkfs.vfat $partition
echo "Mounting partition to first found loop device..."
sudo mkdir -p drive
sudo mount $partition drive
echo "Installing grub to bootable partition..."
sudo grub-install \
--recheck \
--boot-directory=drive/boot/ \
--target=i386-pc \
$device
echo "Creating grub menu config file..."
sudo cat > grub.cfg < EOF
set timeout=30
set default=0
menuentry "Mini OS" {
multiboot /boot/MiniOS.bin
boot
}
EOF
echo "Removing temporary files and resources..."
sudo mv grub.cfg drive/boot/grub/
sudo cp ./src/MiniOS.bin drive/boot/MiniOS.bin
sudo umount drive
sudo rm -r drive
sudo losetup -d $device
echo "Done!"
```

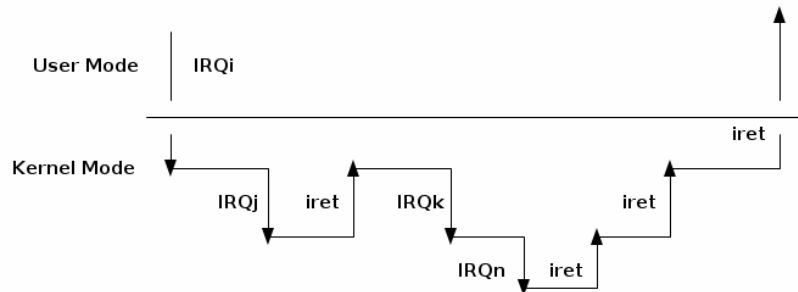
ТЕХНИЧЕСКИЙ ОТЧЁТ ПО ПРАКТИКЕ

1. Создать загрузочный образ миниядра MiniX.
2. Изучить механизм прерывания страничной организации памяти.
3. Составить алгоритм.



Nested interrupts and exceptions

An interrupt handler may preempt both other interrupt handlers and exception handlers. On the other handler never preempts an interrupt handler.



№	Команды	Комментарии
0	free_frame, vmalloc, mem_init, zone_list, die	Форматируем область. Выделяем память с помощью vmalloc. Создаем директорию страниц. Регистрируем прерывание die. Разделяем диск, записывая в регистр CR3 адрес начала директории
1	int, die	Процесс подает сигнал процессору, чтобы сгенерировать прерывание die.
2	pusha, mov ds, ax mov es, ax mov fs, ax mov gs, ax	Сохранение регистров.
3	int, page_fault	Передача управления обработчику прерываний. Обработчик вызывает прерывания page_fault.
4	iret	Возврат из прерывания.
5	pop ebx, mov ds, bx mov es, bx mov fs, bx mov gs, bx	Восстановление регистров.
6	PANIC	Конец работы ОС.

4. Реализовать управление жестким диском.

memory.h

```
#ifndef MEMORY_H
#define MEMORY_H
#include "common.h"

u32int vmalloc_int(u32int sz, int align, u32int *phys);

u32int vmalloc_a(u32int sz);

u32int vmalloc_p(u32int sz, u32int *phys);

u32int vmalloc_ap(u32int sz, u32int *phys);

u32int vmalloc(u32int sz);

#endif // MEMORY_H
```

memory.c

```
#include "memory.h"
#include "monitor.h"

extern u32int end;
u32int placement_address = (u32int)&end;

u32int vmalloc_int(u32int sz, int align, u32int *phys)
{
    if (align == 1 && (placement_address & 0xFFFFF000) )
    {
        placement_address &= 0xFFFFF000;
        placement_address += 0x1000;
    }
    if (phys)
    {
        *phys = placement_address;
    }
    u32int tmp = placement_address;
    placement_address += sz;

    monitor_write("Memory was allocated at: ");
    monitor_write_hex(placement_address);
    monitor_write("\n");
}
```

```

    return tmp;
}

u32int vmalloc_a(u32int sz)
{
    return vmalloc_int(sz, 1, 0);
}

u32int vmalloc_p(u32int sz, u32int *phys)
{
    return vmalloc_int(sz, 0, phys);
}

u32int vmalloc_ap(u32int sz, u32int *phys)
{
    return vmalloc_int(sz, 1, phys);
}

u32int vmalloc(u32int sz)
{
    return vmalloc_int(sz, 0, 0);
}

```

phys_memory.h

```

#ifndef PHYS_MEMORY
#define PHYS_MEMORY

#include "common.h"
#include "isr.h"

typedef struct page
{
    u32int present    : 1;
    u32int rw         : 1;
    u32int user       : 1;
    u32int accessed   : 1;
    u32int unused     : 7;
    u32int frame      : 20;
} page_t;

typedef struct page_table
{
    page_t pages[1024];
} page_table_t;

typedef struct page_directory
{
    page_table_t *tables[1024];
    u32int tablesPhysical[1024];
}

```

```

    u32int physicalAddr;
} page_directory_t;

void mem_init();

void zone_list(page_directory_t *new);

page_t *get_page(u32int address, int make, page_directory_t *dir);

void die(registers_t regs);
void do_page_fault(registers_t regs);

#endif

```

phys_memory.c

```

#include "phys_memory.h"
#include "memory.h"

page_directory_t *kernel_directory=0;

page_directory_t *current_directory=0;

u32int *frames;
u32int nframes;

extern u32int placement_address;

#define INDEX_FROM_BIT(a) (a/(8*4))
#define OFFSET_FROM_BIT(a) (a%(8*4))

static void set_frame(u32int frame_addr)
{
    u32int frame = frame_addr/0x1000;
    u32int idx = INDEX_FROM_BIT(frame);
    u32int off = OFFSET_FROM_BIT(frame);
    frames[idx] |= (0x1 << off);
}

static void clear_frame(u32int frame_addr)
{
    u32int frame = frame_addr/0x1000;
    u32int idx = INDEX_FROM_BIT(frame);
    u32int off = OFFSET_FROM_BIT(frame);
    frames[idx] &= ~(0x1 << off);
}

static u32int test_frame(u32int frame_addr)

```

```

{
    u32int frame = frame_addr/0x1000;
    u32int idx = INDEX_FROM_BIT(frame);
    u32int off = OFFSET_FROM_BIT(frame);
    return (frames[idx] & (0x1 << off));
}

```

```

static u32int first_frame()
{
    u32int i, j;
    for (i = 0; i < INDEX_FROM_BIT(nframes); i++)
    {
        if (frames[i] != 0xFFFFFFFF)    {
            for (j = 0; j < 32; j++)
            {
                u32int toTest = 0x1 << j;
                if ( !(frames[i]&toTest) )
                {
                    return i*4*8+j;
                }
            }
        }
    }
}

```

```

void alloc_frame(page_t *page, int is_kernel, int is_writeable)
{
    if (page->frame != 0)
    {
        return;
    }
    else
    {
        u32int idx = first_frame();
        if (idx == (u32int)-1)
        {
            // ÐµÑ, Ñ·Ð²Ð³Ð±Ð³Ð´Ð½Ñ<Ñ... Ñ,,Ñ€ÐµÐ¹Ð¼Ð²Ð³Ð².
        }
        set_frame(idx*0x1000);
        page->present = 1;
        page->rw = (is_writeable)?1:0;
        page->user = (is_kernel)?0:1;
        page->frame = idx;
    }
}

```

```

void free_frame(page_t *page)
{
    u32int frame;
    if (!(frame=page->frame))
    {
        return;
    }
    else

```

```

    {
        clear_frame(frame);
        page->frame = 0x0;
    }
    monitor_write("\nDISK WAS FORMATED!\n");
}

void mem_init()
{
    u32int mem_end_page = 0x1000000;

    nframes = mem_end_page / 0x1000;
    frames = (u32int*)vmalloc(INDEX_FROM_BIT(nframes));
    memset(frames, 0, INDEX_FROM_BIT(nframes));

    kernel_directory = (page_directory_t*)vmalloc_a(sizeof(page_directory_t));
    current_directory = kernel_directory;

    int i = 0;
    while (i < placement_address)
    {
        alloc_frame( get_page(i, 1, kernel_directory), 0, 0);
        i += 0x1000;
    }
    register_interrupt_handler(14, &die);

    zone_list(kernel_directory);
    monitor_write("\nDISK WAS CREATED!\n");
    switch_user_mode();
}

void switch_user_mode(){
    monitor_write("Switched to user mode\n");
}

void zone_list(page_directory_t *dir)
{
    current_directory = dir;
    asm volatile("mov %0, %%cr3":: "r"(&dir->tablesPhysical));
    u32int cr0;
    asm volatile("mov %%cr0, %0":: "=r"(cr0));
    cr0 |= 0x80000000; // Enable paging!
    asm volatile("mov %0, %%cr0":: "r"(cr0));
}

page_t *get_page(u32int address, int make, page_directory_t *dir)
{
    address /= 0x1000;
    u32int table_idx = address / 1024;
    if (dir->tables[table_idx]) {
        return &dir->tables[table_idx]->pages[address%1024];
    }
    else if(make)
    {

```

```

    u32int tmp;
    dir->tables[table_idx] = (page_table_t*)vmalloc_ap(sizeof(page_table_t), &tmp);
    dir->tablesPhysical[table_idx] = tmp | 0x7; // PRESENT, RW, US.
    return &dir->tables[table_idx]->pages[address%1024];
}
else
{
    return 0;
}
}

```

```

void do_page_fault(registers_t regs)
{
    u32int faulting_address;
    asm volatile("mov %%cr2, %0" : "=r" (faulting_address));

    int present = !(regs.err_code & 0x1);
    int rw = regs.err_code & 0x2;
    int us = regs.err_code & 0x4;
    int reserved = regs.err_code & 0x8;
    int id = regs.err_code & 0x10;

    monitor_write("Page fault! ( ");
    if (present) {monitor_write("present ");}
    if (rw) {monitor_write("read-only ");}
    if (us) {monitor_write("user-mode ");}
    if (reserved) {monitor_write("reserved ");}
    monitor_write("") at 0x";
    monitor_write_hex(faulting_address);
    monitor_write("\n");
    PANIC("Page fault");
}

```

```

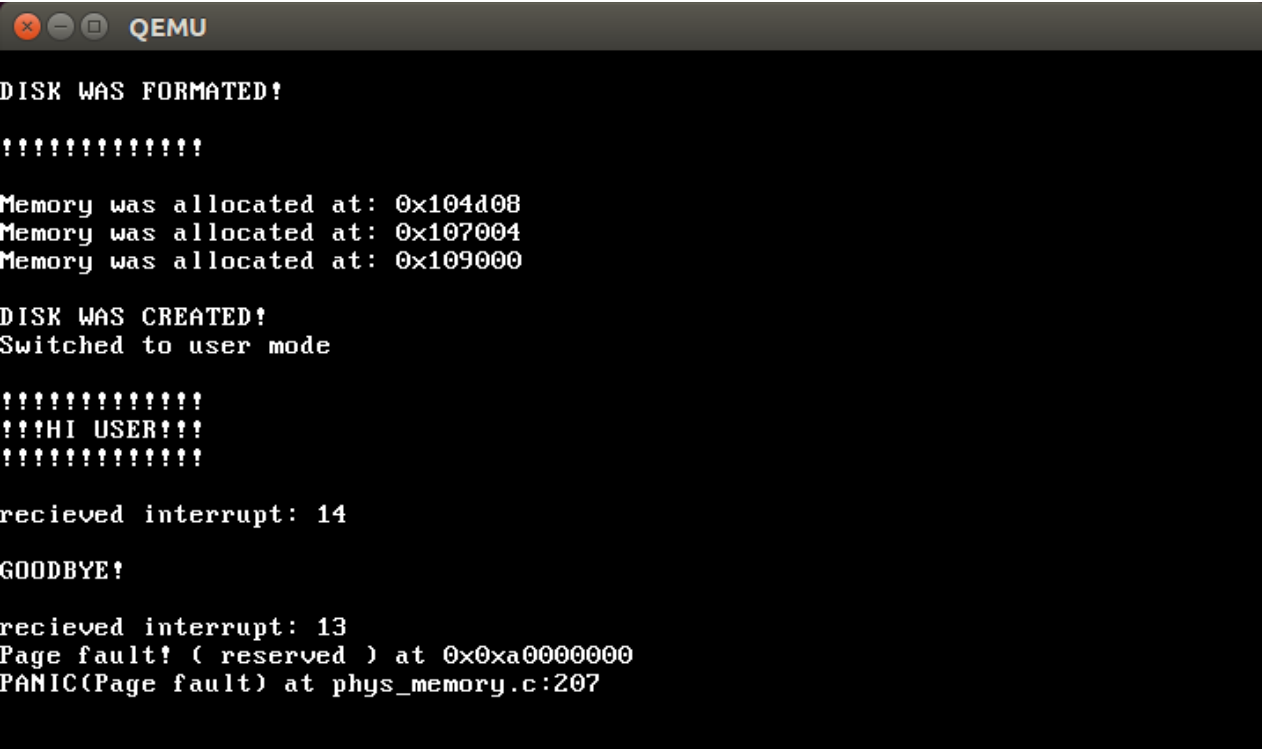
void die(registers_t regs){
    u32int faulting_address;
    asm volatile("mov %%cr2, %0" : "=r" (faulting_address));
    monitor_write("\nGOODBYE!\n\n");
    register_interrupt_handler(13, &do_page_fault);
    asm volatile("int $0xD");
    asm volatile("iret");
}

```

5. Тестирование программы.

Скриншоты работы программы

Папка MiniOS_1. Выделение памяти, инициализация ЖД, прерывания die и do_page_fault.



```
QEMU
DISK WAS FORMATED!
!!!!!!!!!!!!!!
Memory was allocated at: 0x104d08
Memory was allocated at: 0x107004
Memory was allocated at: 0x109000
DISK WAS CREATED!
Switched to user mode
!!!!!!!!!!!!!!
!!!HI USER!!!
!!!!!!!!!!!!!!
recieved interrupt: 14
GOODBYE!
recieved interrupt: 13
Page fault! ( reserved ) at 0x0xa0000000
PANIC(Page fault) at phys_memory.c:207
```

Папка MiniOS_2. Выделение памяти, инициализация ЖД, эмуляция того, что память закончилась, прерывание для выделения дополнительной памяти при ее нехватке, прерывания die и do_page_fault.


```
QEMU
Memory was allocated at: 0x109000

DISK WAS CREATED!
Switched to user mode

!!!!!!!!!!!!!!
!!!HI USER!!!
!!!!!!!!!!!!!!

recieved interrupt: 14

NOT ENOUGH MEMORY

DISK WAS FORMATED!
Memory was allocated at: 0x109100

DISK WAS EXTENDED!
recieved interrupt: 15

GOODBYE!

recieved interrupt: 13
Page fault! ( reserved ) at 0x0xa0000000
PANIC(Page fault) at phys_memory.c:211
```

6. Список используемой литературы.

1. Семенов А.С. «Проектирование сетевых операционных систем. Практический курс»
2. Документация Mini OS
3. Э. Таненбаум, Х. Бос «Современные операционные системы»
4. Карта линукса https://makelinux.net/kernel_map/
5. Функция vmalloc <https://it.wikireading.ru/1876>
6. Таблицы IDT, GDT <http://rus-linux.net/MyLDP/kernel/toyos/sozdaem-unix-like-os-04.html>
7. Страничная организация памяти <http://rus-linux.net/MyLDP/kernel/toyos/sozdaem-unix-like-os-06.html>
8. Прерывания линукс <https://linux-kernel-labs.github.io/master/lectures/interrupts.html>

7. Выводы.

Проделав практическую работу, мы изучили общий механизм прерываний. Освоили некоторые команды языка ассемблер, основные принципы работы процессоров на архитектуре x86, так же разобрались в принципах работы ОС. Реализовали модуль виртуальной памяти и обработку прерывания page_fault, прерывания die, научились выделять дополнительную память при ее нехватке.