|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 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 руководители 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.
2. Изучить механизм прерывания страничной организации памяти.
3. Составить алгоритм.
4. Реализовать управление жестким диском.
5. Тестирование программы.
6. Список используемой литературы.
7. 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:  Отчёт практиканта  1. Создать загрузочный образ миниядра MiniX.  2. Изучить механизм прерывания страничной организации памяти. Построить модель алгоритма.  3. Составить алгоритм.  4. Реализовать управление жестким диском.  5. Тестирование программы.  6. Список используемой литературы.  7. Выводы. | Постановили:  Считать практику выполненной и защищённой на  Ногаев Д.Т. Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Арешин С.О. Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Литвина А.А. Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Ногаев Д.Т. Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Арешин С.О. Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Литвина А.А. Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Ногаев Д.Т. Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Арешин С.О. Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Литвина А.А. Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Ногаев Д.Т. Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Арешин С.О. Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Литвина А.А. Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Ногаев Д.Т. Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Арешин С.О. Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_  Литвина А.А. Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |
|  | Общая оценка ­\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
|  |  |

Руководитель: Семенов А. С. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: 12.07.2019

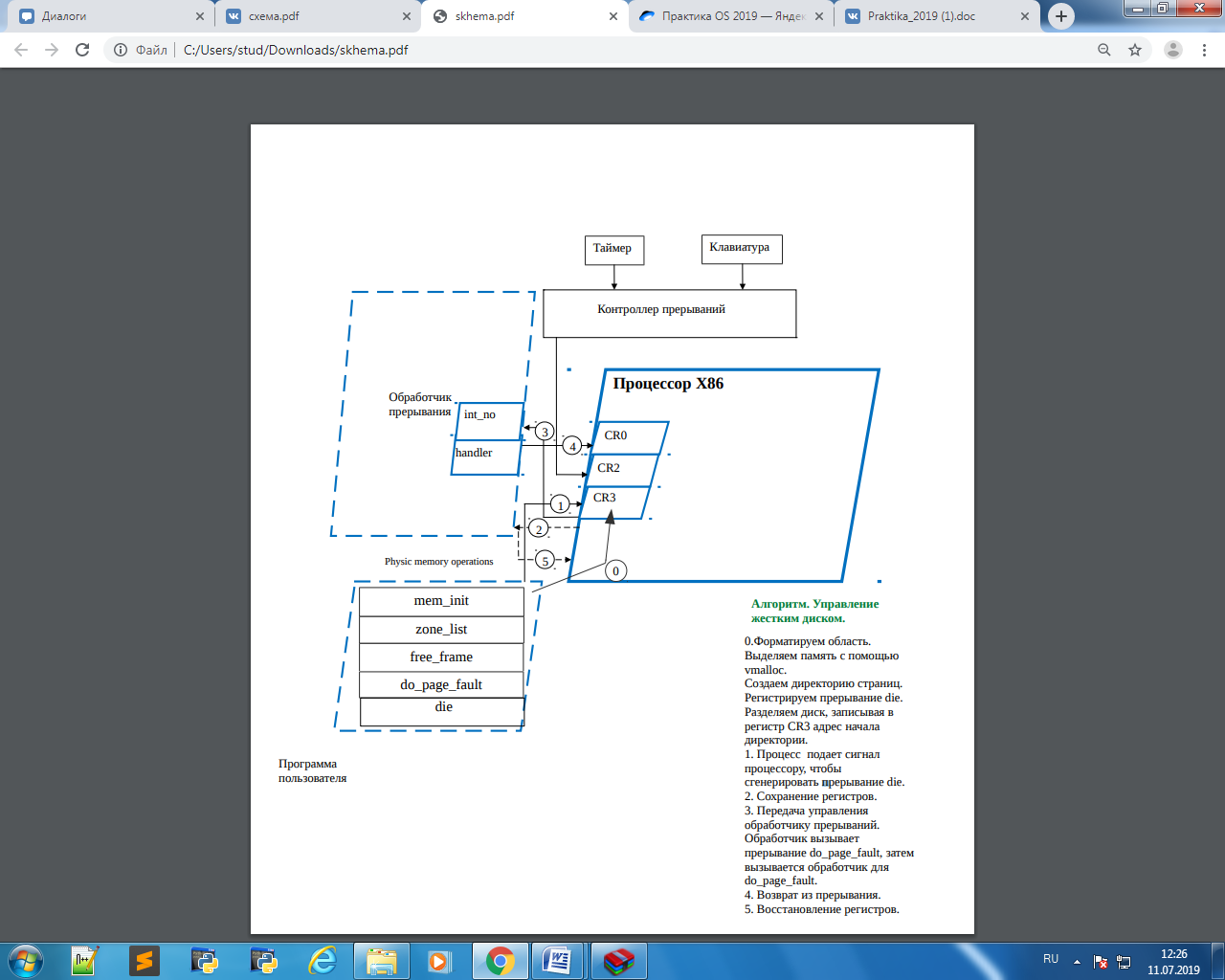
**МАТЕРИАЛЫ ПО РАЦИОНАЛИЗАТОРСКИМ ПРЕДЛОЖЕНИЯМ**

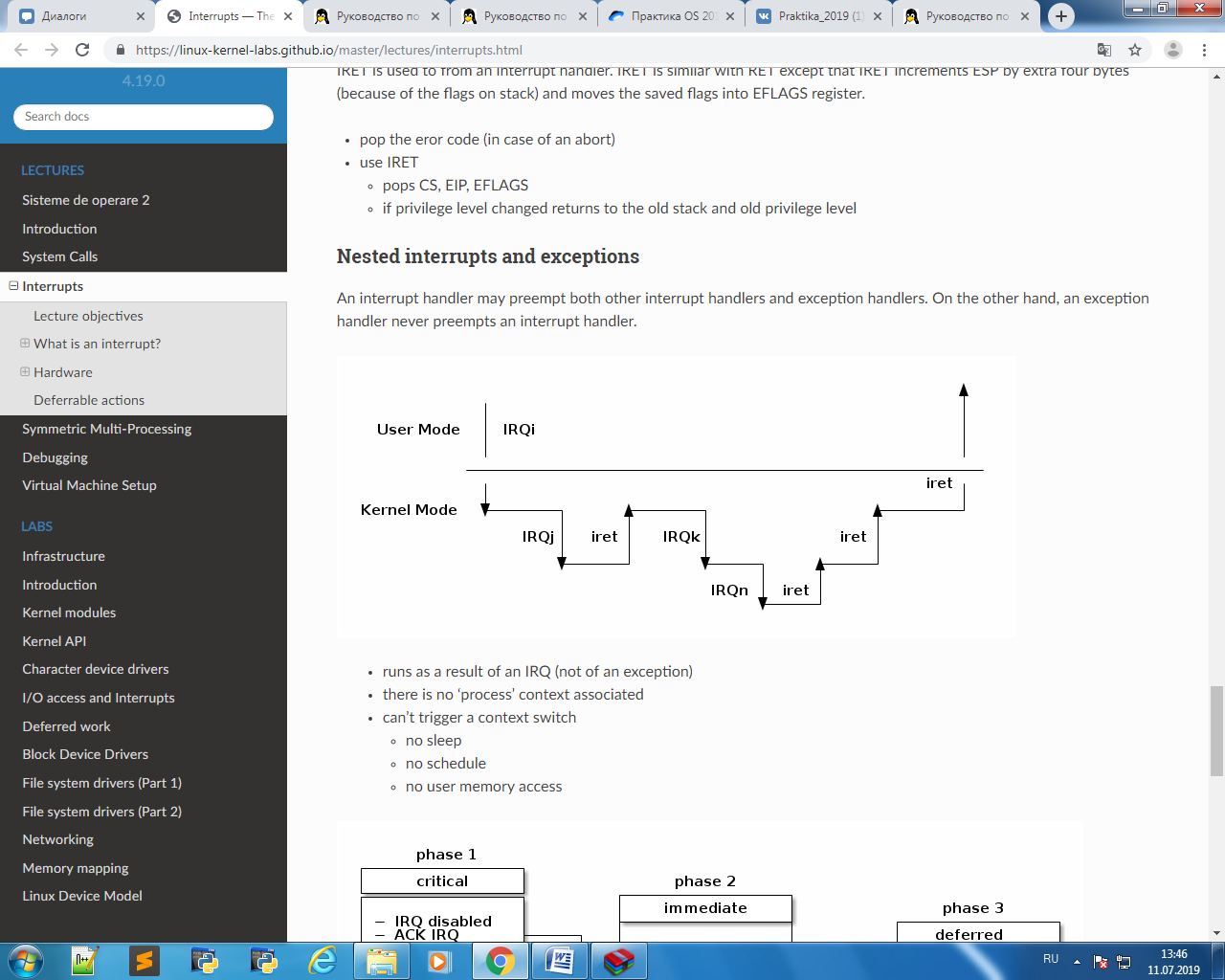
Скрипт для установки образа на новых версиях ОС (Ubuntu 19.04).

#!/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.





|  |  |  |
| --- | --- | --- |
| № | 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. |

* 1. 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)

{

// ÐÐµÑ‚ ÑÐ²Ð¾Ð±Ð¾Ð´Ð½Ñ‹Ñ… Ñ„Ñ€ÐµÐ¹Ð¼Ð¾Ð².

}

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");

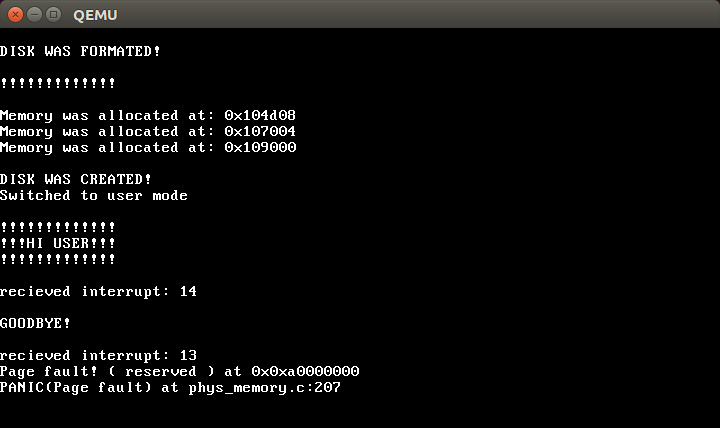
asm volatile("iret");

}

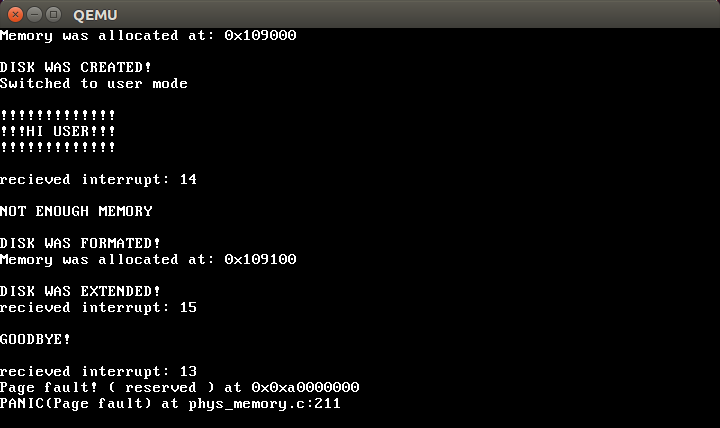
* 1. Тестирование программы.

Скриншоты работы программы

Папка MiniOS\_1. Выделение памяти, инициализация ЖД, прерывания die и do\_page\_fault.



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.



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.