**Министерство образования и науки Российской Федерации**

**федеральное государственное автономное образовательное учреждение высшего образования**

**“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ**

**УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,**

**МЕХАНИКИ И ОПТИКИ”**

**ФАКУЛЬТЕТ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ**

**ОТЧЕТ**

**по Лабораторной работе № 1**

**«РАЗРАБОТКА МИКРО-ОПЕРАЦИОННОЙ СИСТЕМЫ»**

Специальность 09.02.07 «Информационные системы и программирование»

ПМ.01 «Разработка модулей программного обеспечения для компьютерных систем»

МДК.01.04 «Системное программирование»

|  |  |
| --- | --- |
| Преподаватель:  Ошурок Э.Э.  «24» октября 2019г.  Оценка: | Выполнил:  студент группы Y2435  Котлицкий С.А. |

Санкт-Петербург

2019/2020

Цель работы

Получение навыков в области разработки простейших операционных систем, программ загрузки и взаимодействия с оборудованием в защищенном режиме.

индивидуальное задание

Разработать программу загрузчик с возможностью загрузки программы в память и передачи управления последней. Программа должна удовлетворять в обязательном порядке следующим функциям: (1) Наличие анимированной заставки, (2) Возможность завершения работы (3) Наличие способа взаимодействия с пользователем. Реализовать интерфейс пользователя для выполнения арифметических операций над числами (калькулятор) ввод должен происходить при нажатии цифровых клавиш на клавиатуре, отображение должно происходить на экране в специальном поле.

Выполнение

Исходные коды программы представлены далее.

Листинг загрузчика (файл boot.asm):

;32 разрядный защищенный режим работы

org 0x7c00

call readFont

call readProgram

call setVESAMode

cli;отключение возможных прерывание clear interupt - нажатие клавиши в момент запуска режима и тд.

mov sp, 0xfffc;выделили 4 байта памяти в стеке, стек 16-битный

mov ax, 0

call segnull

lgdt[gdt];считывание таблицы - копироване с десрипторную таблицу

;переход в защищенный режим. cr0-системный регистр

mov eax, cr0

or eax, 1; выбор режима работы

mov cr0, eax

mov eax, [buf+0x28];получаем адрес фреймбуфера

mov dword[0x8000],eax

jmp MyCode : 0xB000

ret

readFont:

mov ax, 0x0210

mov dx, 0x0080

mov ch, 00

mov cl, 18

mov bx, 0x8004;8000-адрес фреймбуфера (находится перед \_start())

int 0x13

ret

readProgram:

mov ax, 0x0218

mov dx, 0x0080

mov ch, 00

mov cl, 02

mov bx, 0xB000;8000-адрес фреймбуфера (находится перед \_start())

int 0x13

ret

segnull:;Обнуление сегментных регистров

mov ss, ax

mov ds, ax

mov fs, ax

mov es, ax

mov gs, ax

ret

setVESAMode:;VESA стандарты телеобработки и тд; переключение в продвинутый видеорежим

mov ax, 0x4f01;4f - что-то делать с видоадаптером, 01-получить информацию: взять из видеоадаптера информацию, соотв граф разрешения

mov cx, 0x118 ;1024x768

mov di, buf

int 0x10

mov ax, 0x4f02;переключение видеорежима

mov bx, 0x4118;переключение видеорежима

int 0x10

ret

gd\_table:

Null\_d:

Dd 0

Dd 0

Code\_d:

Dw 0xffff;предел памяти, в которую может расширяться

Dw 0;младшая часть базового адреса

Db 0;адрес

Db 0b10011010; флаги -> p ->dpl(2 бита) -> s -> e -> ed/c -> r/w -> a

;p определяет что сегмент содержит верные код и данные -> сегмент заполнен

;dpl - содержит уровень привилегий, присваемый сегменту

;s - системный дескриптор или дескриптор программы

;e - определяет тип дескриптора (содержит данные или исполняемый код(1))

;ed/c - будет ли происходить расширение сегмента к области данных или к области стека(1)

;r/w - чтение/запись

;a - был ли доступ к дескриптору

;Дескриптор, описывающий свою область кода, содержит правило работы с этим кодом

Db 0b11001111;гранулярность (размер страницы, которые описываются дискриптором)

Db 0;адрес, появилось позже

Data\_d:

Dw 0xffff;65000 кб

Dw 0

Db 0

Db 0b10010010

Db 0b11001111;гранулярность (размер страницы, которые описываются дискриптором)

gd\_table\_end:

gdt:

Dw gd\_table\_end-gd\_table;размер таблицы

Dd gd\_table

MyCode equ Code\_d-gd\_table;присвоение (эквивалент)

;MyData equ Data\_d-gd\_table

buf dd 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 ;104 байта

Листинг кода автоматизированного сборщика:

#!/bin/bash

cd /Users/sergey/Documents/Micro/launch/font-first

nasm -f bin boot.asm -o boot.bin

i686-w64-mingw32-gcc -nostdlib -m32 -masm=intel main.c

i686-w64-mingw32-objcopy -O binary a.exe

pythonw pyt.py

qemu-system-i386 disk.img

Листинг кода операционной системы:

const int fontAdrs = 0x9018; //менять если изменять boot.asm

const int fontCount = 58; //менять если изменять font.c

const int fontWidth = 5; //менять если изменять font.c

const int fontHeight = 6; //менять если изменять font.c

const int pixelWidth = 10; //менять если хотите

const int pixelHeight = 10; //менять если хотите

const int width = 1024; //ширина экрана

const int height = 768; //высота экрана

void drawChar (char, int, int, int); //вывод символа <символ>,<х>,<у>

void drawString (char\*, int, int, int, int, int, int); //вывод символа <строка>,<х>,<у>,<расстояние между букв>, <задержка между выводом>

void drawRect(int, int, int, int, int, int);

void putPixel(int, int, int);

void circle(int x, int y, int r,int color);

void rectangle(int x, int y, int a, int b, int color);

void wait(int time);

int getKey();

int hitKey();

void resetKbd(int key);

int atoi(char\*);

int itoa(int n, char s[]);

int start() {

char str[] = "SHAPE";

int i,j;

// x = 512 y = 384 a = 20 b = 30 startX = 502 startY = 369

for(i=0; i < 3000;i+=40) {

circle(0, i, i, 0xff0000);

wait(10000);

}

for(i=0; i < 3000;i+=40) {

circle(0, i, i, 0x000000);

wait(9000);

}

int size = 15;

for(i=0; i < 400; i+=size) {

rectangle(502-2\*i/3, 369-i, 20+5\*i/4, 30+2\*i, 0xffffff);

if(i > 40) {

rectangle(502-2\*(i-size\*5)/3, 369-(i-size\*5), 20+5\*(i-size\*5)/4, 30+2\*(i-size\*5), 0x000000);

}

wait(50000);

}

for(i=0; i<25; i+=5) circle(400+i, 394, 100, 0x00ff00);

for(i=0; i<100;i+=5) {

circle(550+i, 300, 20, 0x00ff00);

circle(550+i, 400, 20, 0x00ff00);

circle(550+i, 500, 20, 0x00ff00);

circle(550, 300+i, 20, 0x00ff00);

circle(650, 400+i, 20, 0x00ff00);

wait(10000);

}

drawString(str, 330, 120, 20, 0, 5, 10);

wait(900000);

drawRect(0, 0, width, height, 0x000000, 1);

rectangle(212, 95, 600, 60, 0xffffff);

rectangle(212, 175, 600, 60, 0xffffff);

rectangle(212, 295, 600, 60, 0xffffff);

int len = 0;

int line = 110;

int A = 0;

int B = 0;

int menu = 0;

int resV = 0;

int resLen = 0;

char text[] = " ";

char result[] = " ";

char res[] = "RES";

char enter[] = "ENTER TO CALC";

char esc[] = "ESC TO EXIT";

char bs[] = "BACKSPACE TO RESET";

char reset[] = "R TO REBOOT";

int minusA = 0;

int minusB = 0;

int minusRes = 0;

drawChar('A', 155, 110, 5);

drawChar('B', 155, 190, 5);

drawString(res, 110, 315, 5, 0, 3, 5);

drawRect(217, 470, 30, 3, 0xffffff, 1);//plus

drawRect(230, 457, 3, 30, 0xffffff, 1);//plus

drawRect(267, 470, 30, 3, 0xffffff, 1);//minus

for(i=0; i < 10; i++){

drawRect(317+i\*3, 485-i\*3, 3, 3, 0xffffff, 1);//mul

drawRect(317+i\*3, 458+i\*3, 3, 3, 0xffffff, 1);//mul

drawRect(367+i\*3, 485-i\*3, 3, 3, 0xffffff, 1);//div

}

drawString(enter, 215, 530, 5, 0, 13, 5);

drawString(esc, 215, 570, 5, 0, 11, 5);

drawString(bs, 215, 610, 5, 0, 18, 5);

drawString(reset, 215, 650, 5, 0, 11, 5);

int btn;

while(1){

btn = getKey();

if(btn == 0x01) { // esc btn

break;

}

if(btn == 0x13) { // R btn

asm("sti");

resetKbd(0x13);

}

if(btn == 0x0c) { // minus btn

if (line == 110) {

minusA = 1 - minusA;

}

if (line == 190) {

minusB = 1 - minusB;

}

}

if(btn == 0x1c) { // enter btn

minusRes = 0;

drawRect(215, 305, 500, 50, 0x000000, 1);

resV = 0;

resLen = 0;

if(minusA && A > 0) {

A \*= -1;

}

if(minusB && B > 0) {

B \*= -1;

}

if(menu == 0) {

resV = A + B;

}

if(menu == 1) {

resV = A - B;

}

if(menu == 2) {

resV = A \* B;

}

if(menu == 3) {

if(B != 0) {

resV = A / B;

}

}

if(!resV) {

result[0] = '0';

drawString(result, 250, 310, 5, 100, 1, 5);

}

if(resV < 0) {

minusRes = 1;

resV \*= -1;

}

resLen = itoa(resV, result);

drawString(result, 250, 310, 5, 100, resLen, 5);

}

if (len < 7) {

if(btn == 0x02) {

text[len] = '1';

len++;

}

if(btn == 0x03) {

text[len] = '2';

len++;

}

if(btn == 0x04) {

text[len] = '3';

len++;

}

if(btn == 0x05) {

text[len] = '4';

len++;

}

if(btn == 0x06) {

text[len] = '5';

len++;

}

if(btn == 0x07) {

text[len] = '6';

len++;

}

if(btn == 0x08) {

text[len] = '7';

len++;

}

if(btn == 0x09) {

text[len] = '8';

len++;

}

if(btn == 0x0A) {

text[len] = '9';

len++;

}

if(btn == 0x0B) {

text[len] = '0';

len++;

}

if(btn == 0x34) {

text[len] = 'D';

len++;

}

}

if(btn == 0x50) {// Кнопка вниз

if(line == 110) {

A = atoi(text);

for(i = 0; i < 10; i++) {

text[i] = " ";

}

len = itoa(B, text);

line = 190;

} else {

B = atoi(text);

for(i = 0; i < 10; i++) {

text[i] = " ";

}

len = itoa(A, text);

line = 110;

}

}

if(btn == 0x48) {// Кнопка вверх

if(line == 110) {

A = atoi(text);

for(i = 0; i < 10; i++) {

text[i] = " ";

}

len = itoa(B, text);

line = 190;

} else {

B = atoi(text);

for(i = 0; i < 10; i++) {

text[i] = " ";

}

len = itoa(A, text);

line = 110;

}

}

if(btn == 0x4D) {// Кнопка вправо

if(menu == 3) {

menu = 0;

} else {

menu++;

}

}

if(btn == 0x4B) {// Кнопка влево

if(menu == 0) {

menu = 3;

} else {

menu--;

}

}

if(btn == 0x0e) { // backspace btn

drawRect(215, 105, 500, 50, 0x000000, 1);

drawRect(215, 185, 500, 50, 0x000000, 1);

drawRect(215, 305, 500, 50, 0x000000, 1);

A = 0;

B = 0;

len = 0;

minusA = 0;

minusB = 0;

minusRes = 0;

for(i = 0; i < 10; i++) {

text[i] = " ";

}

}

resetKbd(btn);

drawRect(150, 150, 40, 5, 0x000000, 1);

drawRect(150, 230, 40, 5, 0x000000, 1);

drawRect(150, line+40, 40, 5, 0xffffff, 1);

drawRect(212, 500, 40+3\*50, 5, 0x000000, 1);

drawRect(212+menu\*50, 500, 40, 5, 0xffffff, 1);

drawString(text, 250, line, 5, 100, len, 5);

drawRect(220, 125, 20, 5, 0x000000, 1);

drawRect(220, 205, 20, 5, 0x000000, 1);

drawRect(220, 320, 20, 5, 0x000000, 1);

if (minusA)

drawRect(220, 125, 20, 5, 0xffffff, 2);

if (minusB)

drawRect(220, 205, 20, 5, 0xffffff, 2);

if (minusRes)

drawRect(220, 320, 20, 5, 0xffffff, 2);

wait(100000);

}

drawRect(0, 0, 1024, 768, 0x000000, 1);

char exit[] = "THE POWER CAN BE TURNED OFF";

drawString(exit, 100, 350, 5, 100, 27, 5);

while(1);

return 0;

}

void drawString (char \*arr, int x, int y, int space, int wait\_t, int length, int size) {

int i = 0;

while(i<length) {

drawChar(\*arr, x, y, size);

x += space+fontWidth\*size;

i++;

arr++;

wait(wait\_t);

}

}

void drawChar (char ch, int x, int y, int size) {

int pixelHeight = size, pixelWidth = size;

int positionChar;

char \*p = fontAdrs;

if( (int)ch == 65) positionChar = 0;

if( (int)ch == 66) positionChar = 1;

if( (int)ch == 67) positionChar = 2;

if( (int)ch == 68) positionChar = 3;

if( (int)ch == 69) positionChar = 4;

if( (int)ch == 70) positionChar = 5;

if( (int)ch == 71) positionChar = 6;

if( (int)ch == 72) positionChar = 7;

if( (int)ch == 73) positionChar = 8;

if( (int)ch == 74) positionChar = 8;

if( (int)ch == 75) positionChar = 10;

if( (int)ch == 76) positionChar = 11;

if( (int)ch == 77) positionChar = 12;

if( (int)ch == 78) positionChar = 13;

if( (int)ch == 79) positionChar = 14;

if( (int)ch == 80) positionChar = 15;

if( (int)ch == 81) positionChar = 16;

if( (int)ch == 82) positionChar = 17;

if( (int)ch == 83) positionChar = 18;

if( (int)ch == 84) positionChar = 19;

if( (int)ch == 85) positionChar = 20;

if( (int)ch == 86) positionChar = 21;

if( (int)ch == 87) positionChar = 22;

if( (int)ch == 88) positionChar = 23;

if( (int)ch == 89) positionChar = 24;

if( (int)ch == 90) positionChar = 25;

if( (int)ch == 49) positionChar = 60;

if( (int)ch == 50) positionChar = 61;

if( (int)ch == 51) positionChar = 62;

if( (int)ch == 52) positionChar = 63;

if( (int)ch == 53) positionChar = 64;

if( (int)ch == 54) positionChar = 65;

if( (int)ch == 55) positionChar = 66;

if( (int)ch == 56) positionChar = 67;

if( (int)ch == 57) positionChar = 68;

if( (int)ch == 48) positionChar = 59;

if( (int)ch == 32) positionChar = 69;

for(int i=0; i<positionChar; i++) p += fontWidth\*fontHeight;

for(int i=y; i<y+fontHeight\*pixelHeight; i+=pixelHeight) {

for(int j=x; j<x+fontWidth\*pixelWidth; j+=pixelWidth) {

if (\*p == 1){

drawRect(j,i,pixelWidth,pixelHeight, 0xffffff, 2);

}

p++;

}

}

}

void drawRect(int x, int y, int pixelWidth, int pixelHeight, int color, int step) {

for(int i=y; i<y+pixelHeight; i+=step)

for(int j=x; j<x+pixelWidth; j++)

putPixel(j,i,color);

}

void putPixel(int x,int y,int color){

int \*p = 0x8000;

char \*c=\*p;

c+=((y\*1024)+x)\*3;

\*(int\*)c=color;

}

void circle(int x, int y, int r,int color){ //X,Y - координаты центра, R - радиус, color-цвет

int xd = 0;

int yd = r;

int delta = 1 - 2 \* r;

int error = 0;

while (yd >= 0){

putPixel(x + xd, y + yd, color);

putPixel(x + xd, y - yd, color);

putPixel(x - xd, y + yd, color);

putPixel(x - xd, y - yd, color);

error = 2 \* (delta + yd) - 1;

if ((delta < 0) && (error <= 0)){

delta += 2 \* ++xd + 1;

continue;

}

if ((delta > 0) && (error > 0)){

delta -= 2 \* --yd + 1;

continue;

}

delta += 2 \* (++xd - yd--);

}

}

void wait(int time){

for(int i=0; i<time\*100; i++);

}

void rectangle(int x, int y, int a, int b, int color) {

int i;

for(i=x; i<x+a;i++) {

putPixel(i, y, color);

putPixel(i, y+b, color);

}

for(i=y; i<y+b;i++) {

putPixel(x, i, color);

putPixel(x+a, i, color);

}

}

int getKey() {

asm("mov eax, 0");

asm("in al, 0x60");

}

void resetKbd(int key) {

asm("mov eax, 0");

asm("mov al, [ebp + 4]");

asm("out 0x64, al");

}

int atoi(char\* s){

int n = 0;

while( \*s >= '0' && \*s <= '9' ) {

n \*= 10;

n += \*s++;

n -= '0';

}

return n;

}

int itoa(int n, char s[]){

if(n == 0) {

return 0;

}

int i, j, len=0;

i = 0;

do { /\* генерируем цифры в обратном порядке \*/

s[i++] = n % 10 + '0'; /\* берем следующую цифру \*/

len++;

} while ((n /= 10) > 0); /\* удаляем \*/

s[i] = '\0';

char c;

for (i = 0, j = len-1; i<j; i++, j--) {

c = s[i];

s[i] = s[j];

s[j] = c;

}

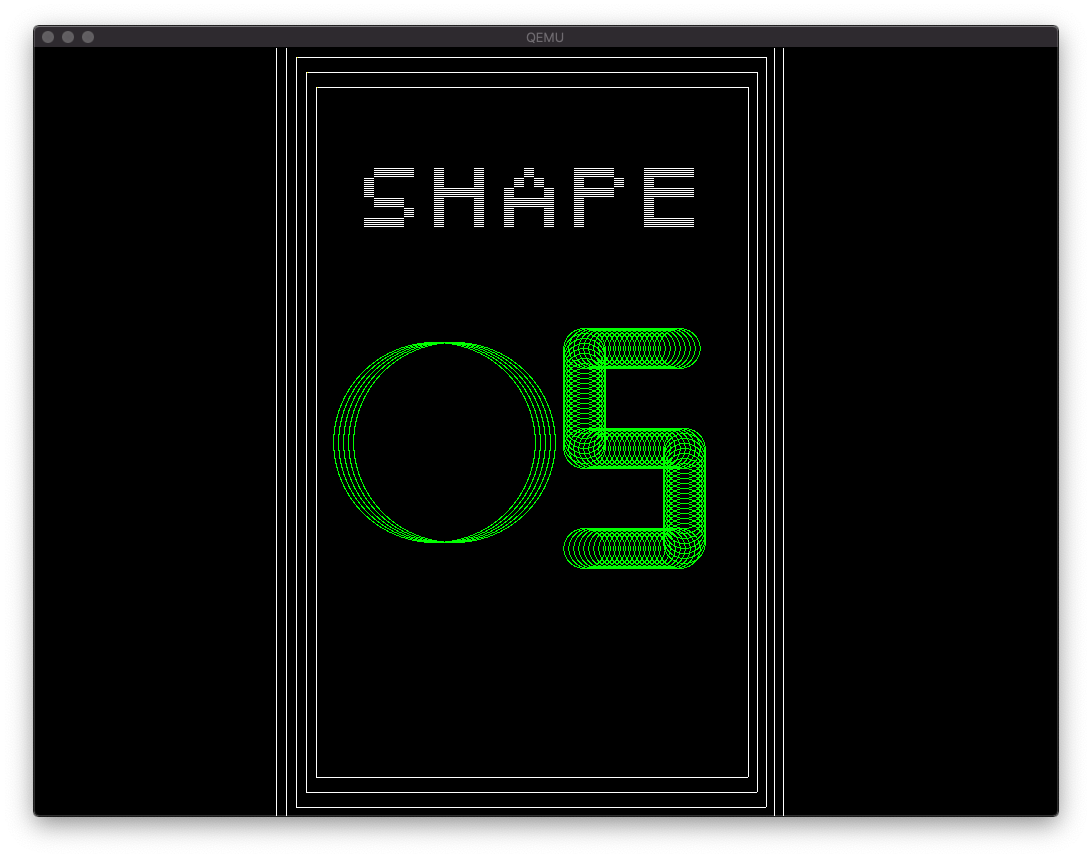
return len;

}

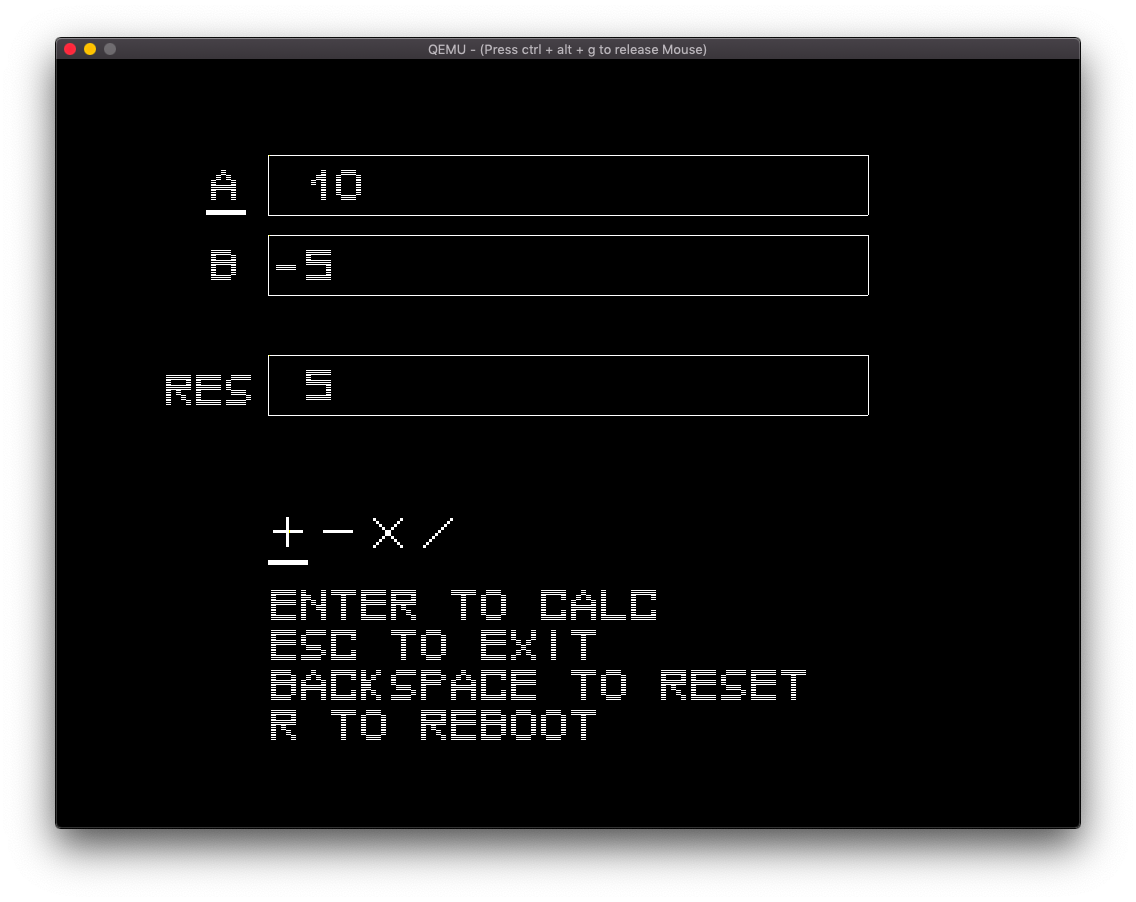
Интерфейсы ОС:

На рисунке 1 представлен загрузочный экран ОС.  
На рисунках 2-5 представлен интерфейс ОС, а также продемонстрированы результаты работы калькулятора.

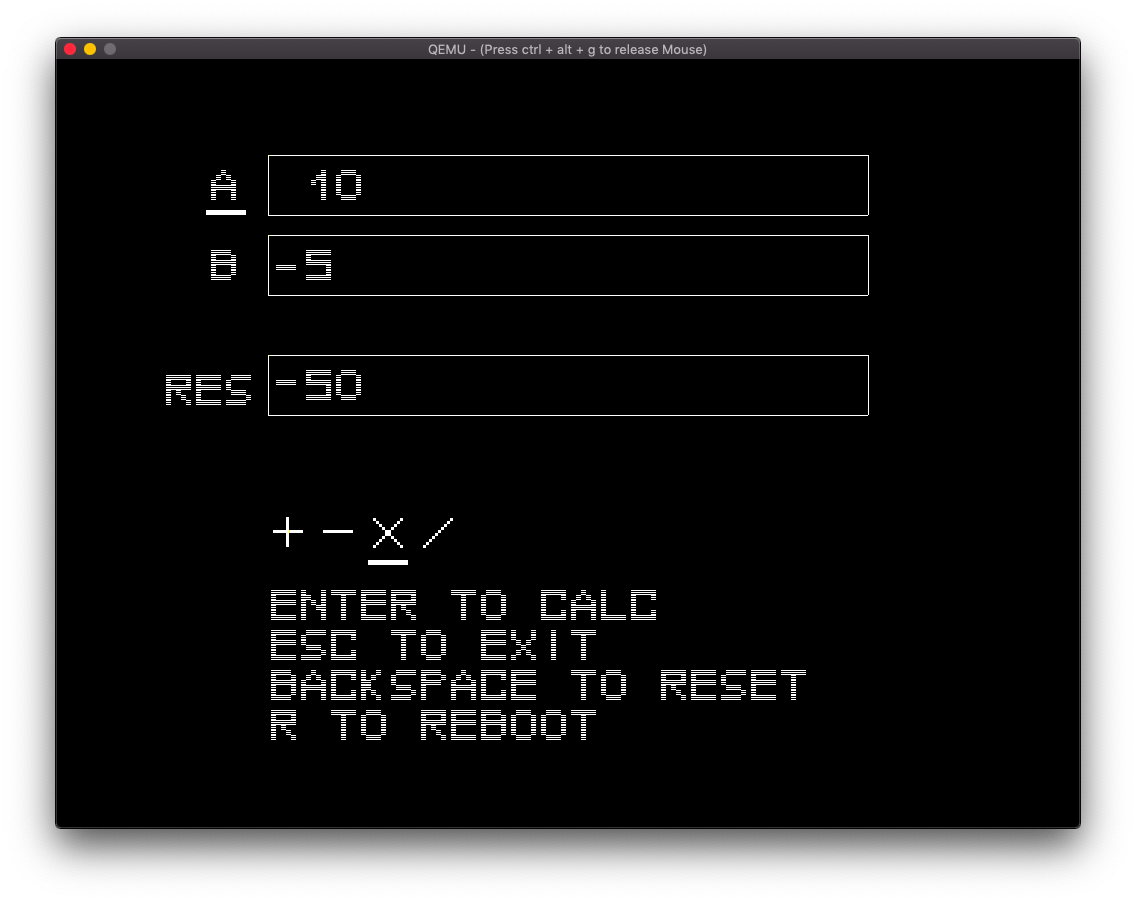
На рисунке 6 представлен интерфейс завершения работы ОС.



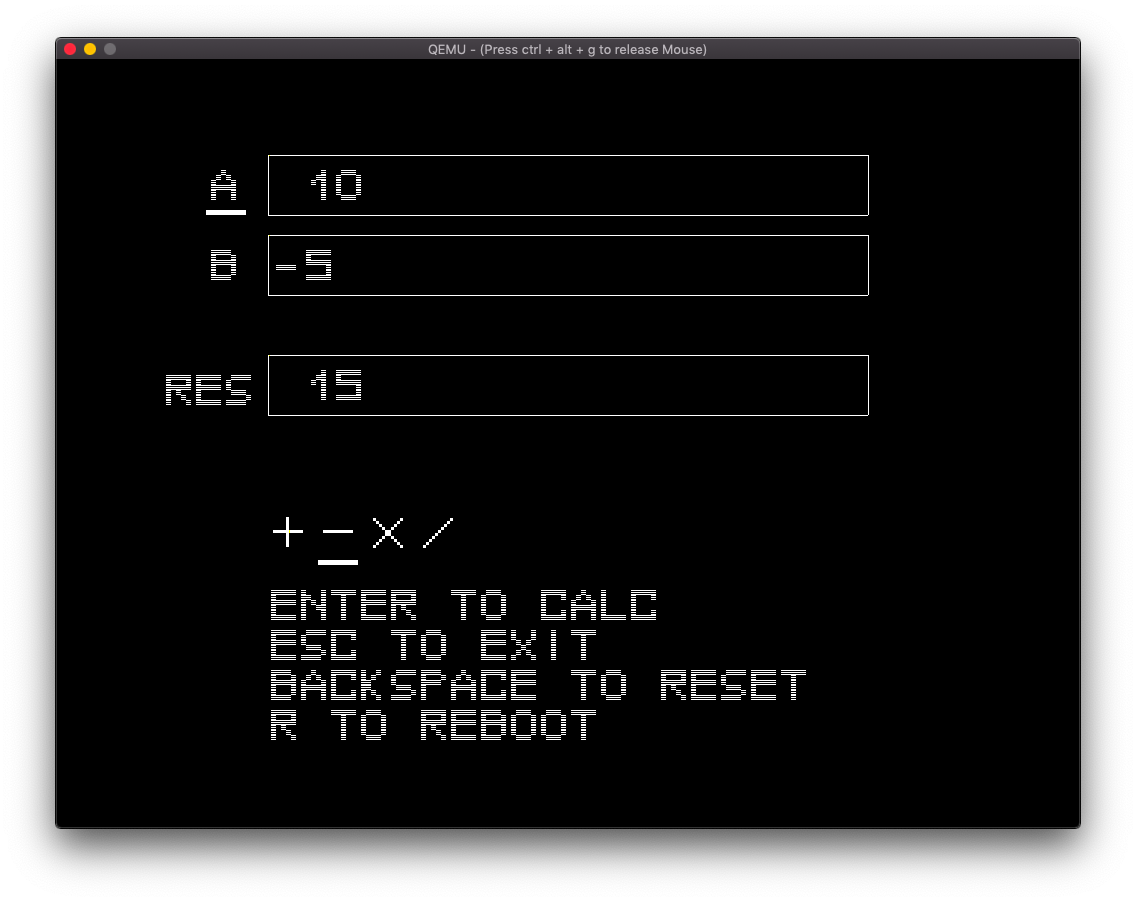
*Рисунок 1 Загрузочный экран*



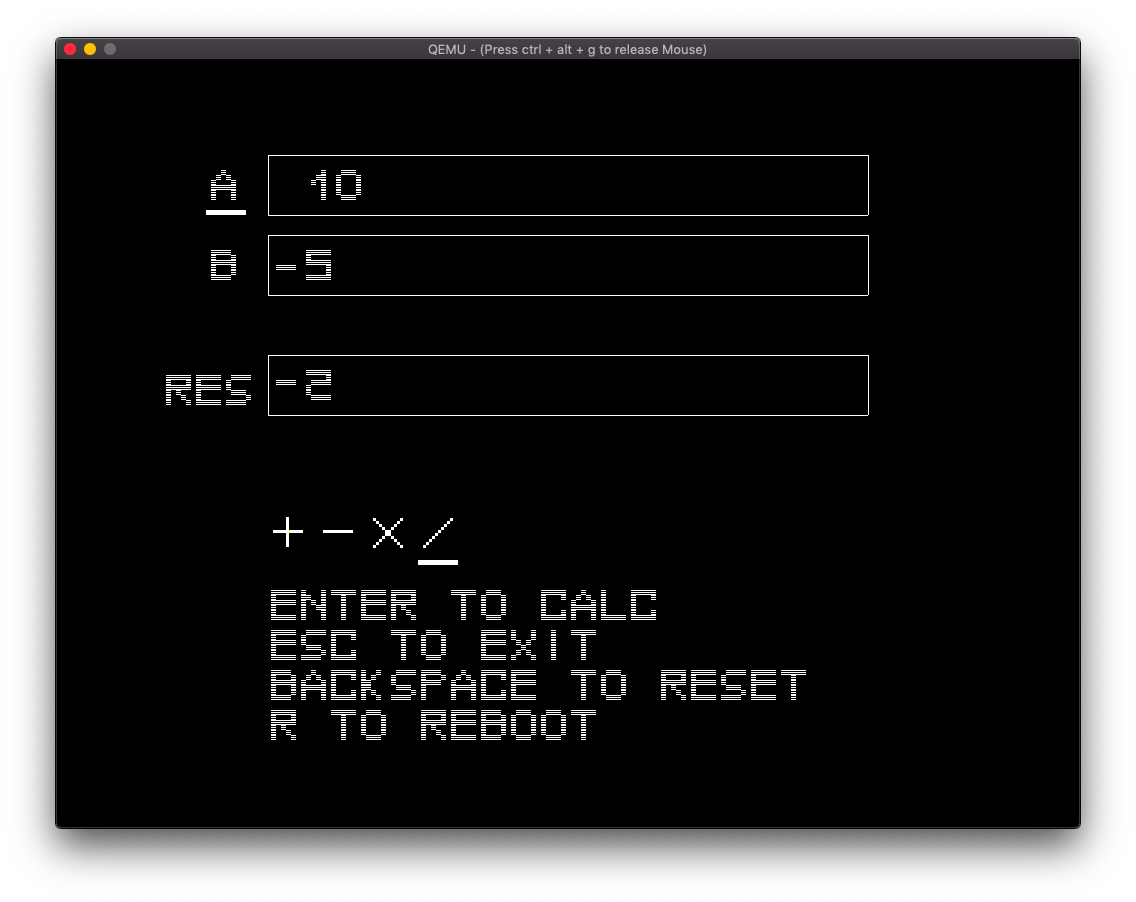
*Рисунок 2 Операция сложения*

**

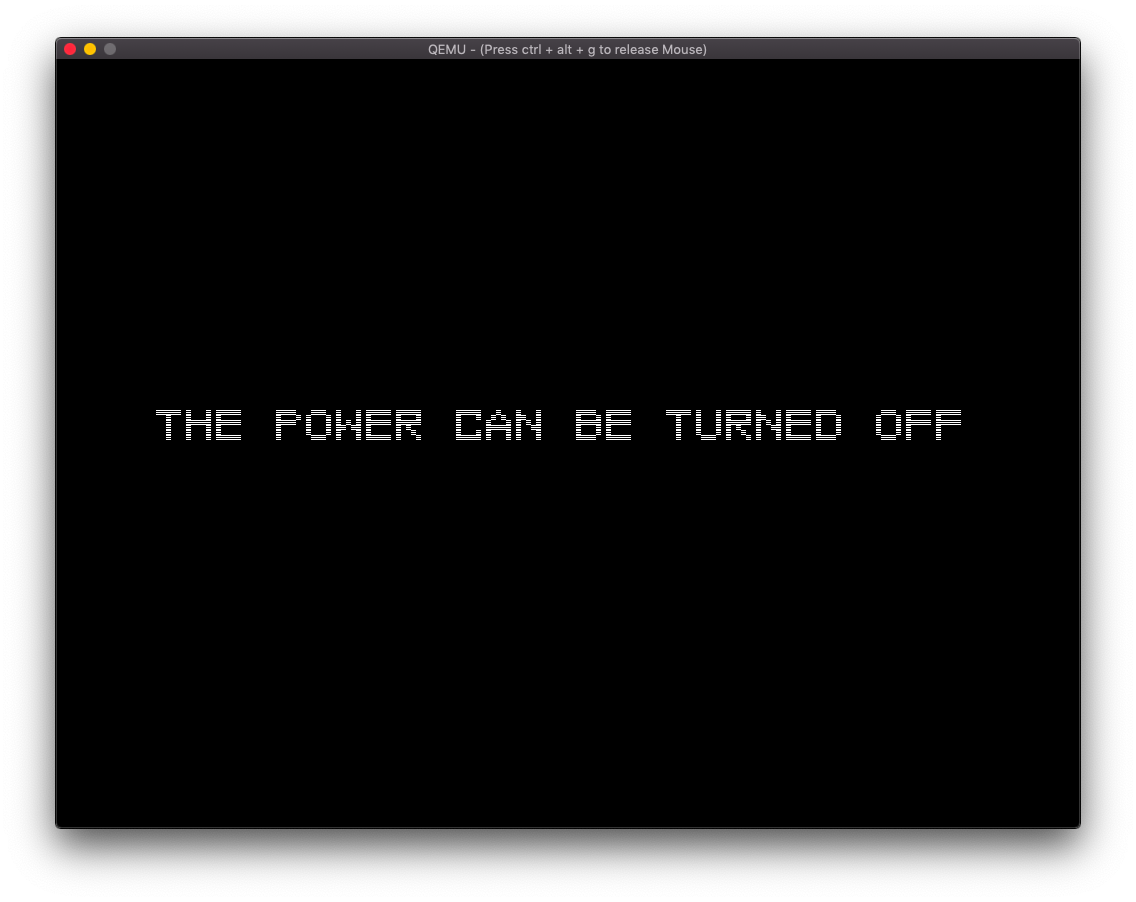
*Рисунок 3 Операция умножения*



*Рисунок 4 Операция вычитания*



*Рисунок 5 Операция деления*



*Рисунок 6 Выход из ОС*