Федеральное агентство связи Федеральное государственное бюджетное образовательное учреждение высшего образования

«Сибирский государственный университет телекоммуникаций и информатики»

Факультет: Информатики и вычислительной техники

Кафедра телекоммуникационных сетей и вычислительных средств

Дисциплина: Архитектура ЭВМ

Отчёт по лабораторной работе №6

«Подсистема прерываний ЭВМ. Сигналы и их обработка»

Выполнили студенты группы ИА-831:

Зарубин Максим Евгеньевич

Дорощук Никита Андреевич

Проверил преподаватель:

Токмашева Елизавета Ивановна

Новосибирск

2020

Задание к лабораторной работе.

1. Прочитайте главу 6 пособия по курсу «ЭВМ и периферийные устройства». Изучите страницу man для функций signal, setitimer.
2. Доработайте консоль Simple Computer. Создайте обработчик прерываний от системного таймера так, чтобы при каждом его срабатывании при нулевом значении флага «игнорирование тактовых импульсов» значение регистра «instructionCounter» увеличивалось на 1, а при поступлении сигнала SIGUSR1 состояние Simple Computer возвращалось в исходное. Обработка нажатых клавиш осуществляется только в случае, если сигналы от таймера не игнорируются.
3. Защита лабораторной работы
   Для защиты лабораторной работы необходимо подготовить программу, реализующую консоль управления Simple Computer и демонстрирующую работу обработчика прерываний.

Описание реализованных функций.

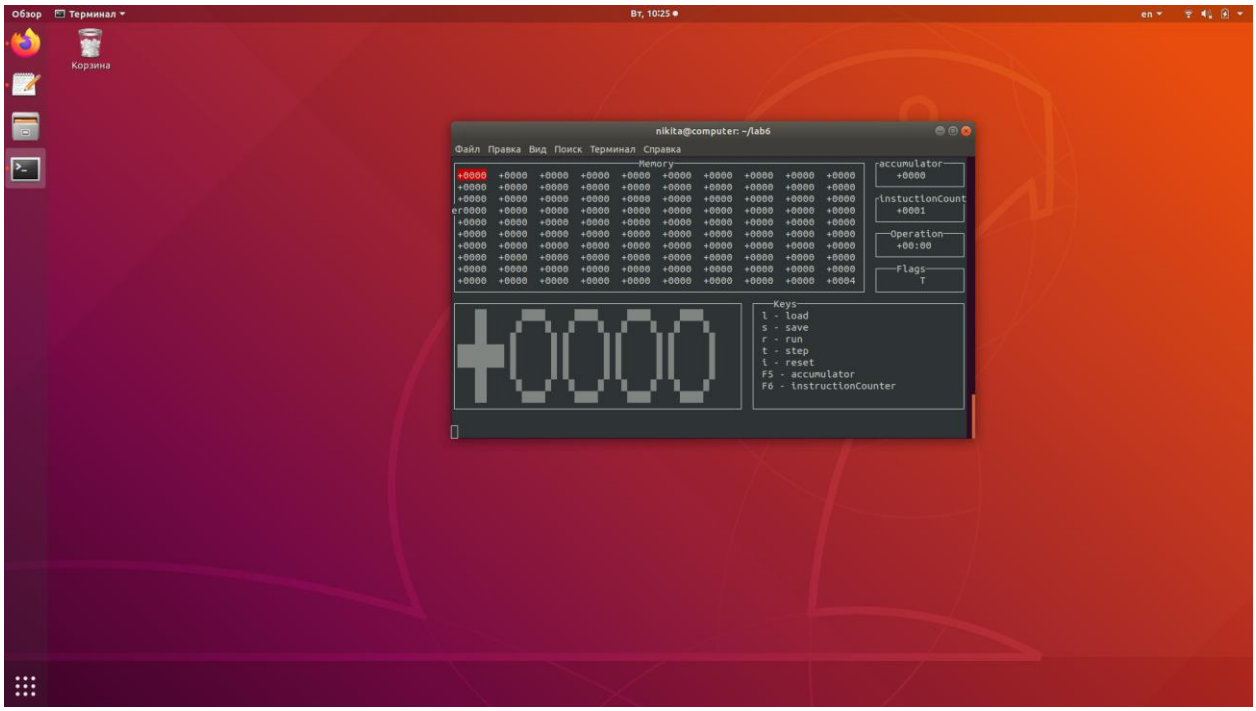
void timerStart () – Используем функцию setitimer для установки таймера.

void setTimerValues (int upperValue, int lowerValue) – Устанавливаем параметры для таймера с помощью структур itimerval и timeval.

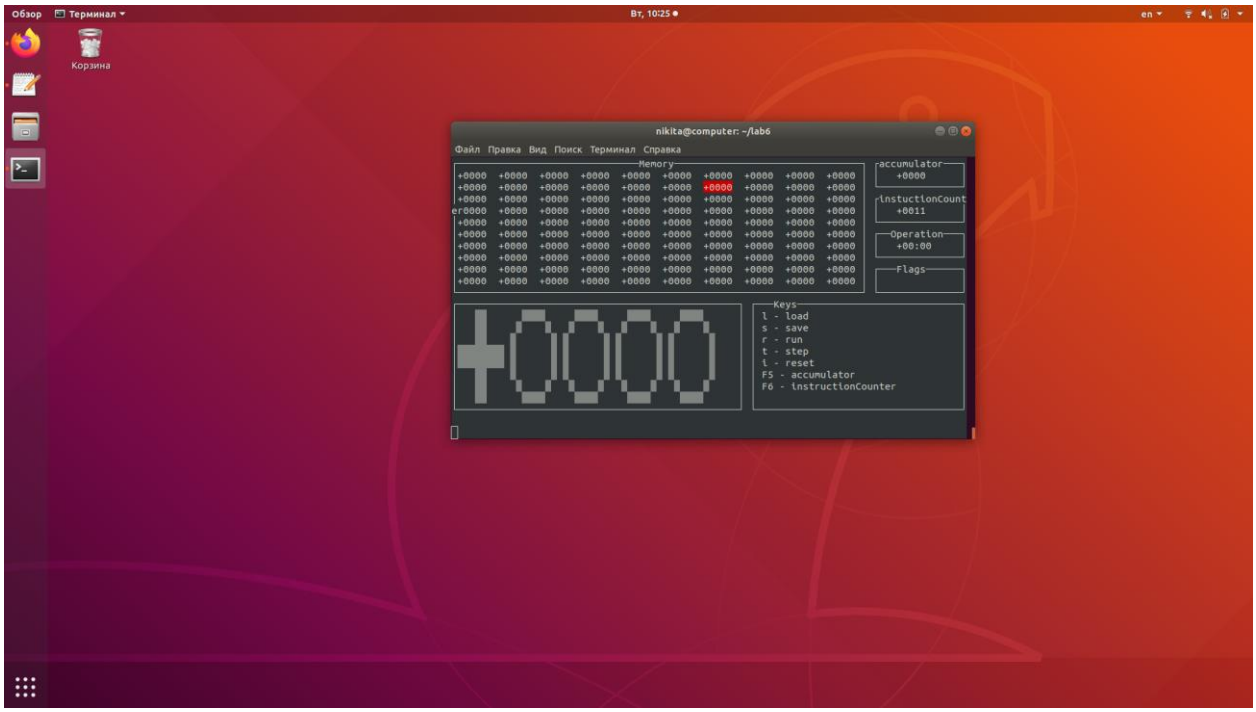
void signalHandler (int signo) – Получаем значения флагов ISRUN и IGNOREFLAG с помощью функции sc_regGet. Проверяем значения с помощью цикла и если проходит, то проверяем, чтобы значение ячейки было нулевым и декодирования значения тоже было равно нулю, тогда уже увеличиваем значение на один, иначе возвращаем значение в исходное и выключаем таймер, это же проделываем если наши значения ошибочные при проверки циклом. И зануляем значение флага ISRUN.

Скриншоты проверки работы функций.

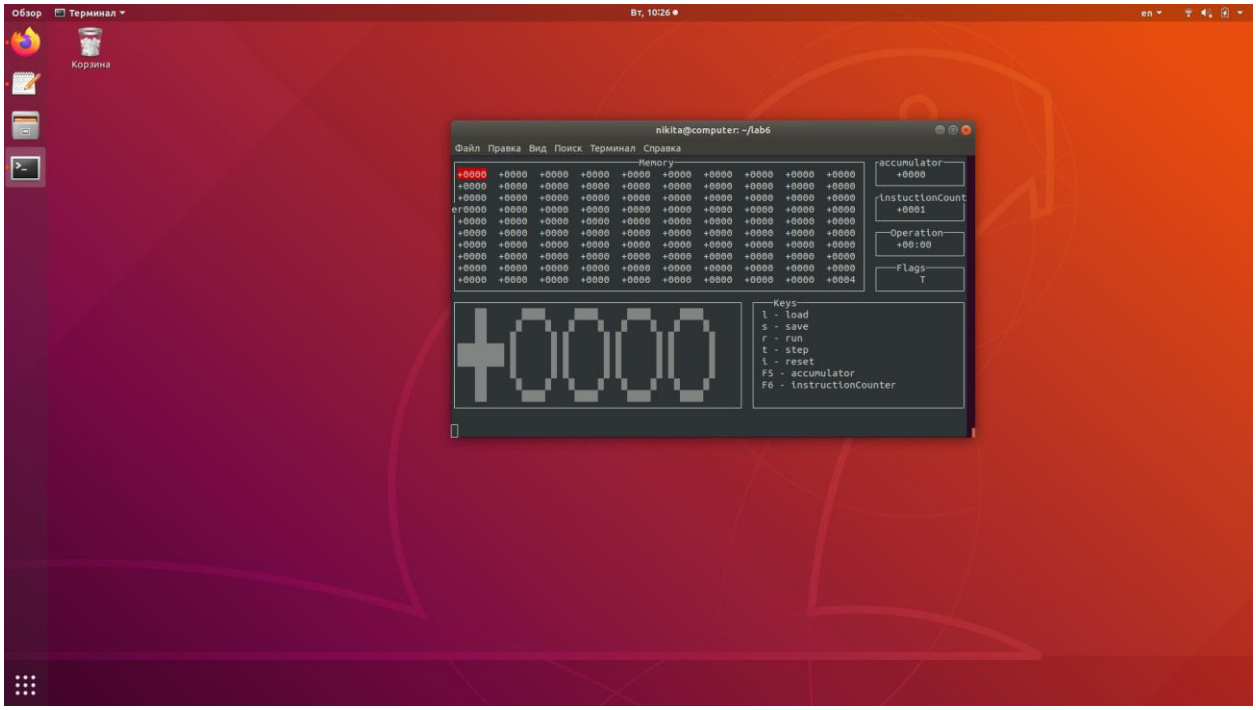
Активируем таймер нажатием клавиши ‘r’



Курсор двигается вправо, игнорируя нажатие клавиш



После нажатия клавиши ‘i’ курсор возвращается в первую ячейку



Листинг программы.

```
#include <sys/types.h>
#include <stdio.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <termios.h>
#include <sys/time.h>
#include <signal.h>
#include <unistd.h>
#include "myterminal.h"
#include "bigchar.h"
#include "memorylib.h"
#include "My_ReadKey.h"
#define P 0
#define O 1
#define M 2
#define T 3
#define E 4

int iC = 1;
int status = 1,t2 = 0;
int statusAc = 1, t1 = 0;
int statusIc = 1;
char temp[4], temp1[4];

int verifMemory(int t){
	int op, com;
	sc_commandDecode(t,&com,&op);
	int mass[40] =
{0x10,0x11,0x20,0x21,0x30,0x31,0x32,0x33,0x40,0x41,0x42,0x43,0x51,0x52,0x53,0x54,0x55,0x5
6,0x57,0x58,0x59,0x60,0x61,0x62,
0x63,0x64,0x65,0x66,0x67,0x68,0x69,0x70,0x71,0x72,0x73,0x74,0x75,0x76};
	int status = -1;
	int i;
	for(i = 0; i < 40; i++){
		if(2*mass[i] == com){
			status = 0;
```

```
                        break;
                }
        }
        return status;
}

void draw(){
        mt_clrscr();
        bc_box(0,0,13,72);
        bc_box(0,73,4,75);
        bc_box(4,73,3,75);
        bc_box(7,73,3,75);
        bc_box(10,73,3,75);
        bc_box(13,52,10,75);
        bc_box(13,0,10,51);
        mt_gotoXY(1,4*7+5);
        write(1,"Memory",6);
        int i,j = 0,value,k=2;
        for(i = 1; i <= 100; i++,j++){
                sc_memoryGet(i,&value);
                if(i == iC && statusIc == 1){
                        mt_setbgcolor(RED);
                }
                mt_gotoXY(k,j*7+2);
                char buf1[8];
                int l1 = sprintf(buf1,"+%.4x",value);
                write(1,buf1,5);
                if(i%10==0){
                        k++;
                        j = -1;
                }
                if(i == iC && statusIc == 1){
                        mt_setbgcolor(BLACK);
                }
        }

        mt_gotoXY(1,11*7-3);
        write(1,"accumulator",11);
        mt_gotoXY(2,11*7);
        if(statusAc == 1){
                sc_memoryGet(iC,&value);
                char buf3[8];
                int l3 = sprintf(buf3,"+%.4x",value);
                write(1,buf3,l3);
        } else if(statusAc == 0){
                char buf3[8];
                int l3 = sprintf(buf3,"+%.4x",t1);
                write(1,buf3,l3);
        }
        mt_gotoXY(4,11*7-3);
        write(1,"instuctionCounter",17);
        mt_gotoXY(5,11*7);
        sc_memoryGet(iC,&value);
        char buf2[8];
        int l2 = sprintf(buf2,"+%.4x",iC);
        write(1,buf2,l2);
        mt_gotoXY(7,11*7-1);
        write(1,"Operation",9);
        mt_gotoXY(8,11*7);
        write(1,"+00:00",6);
        mt_gotoXY(10,11*7);
        write(1,"Flags",5);
        int reg;
        sc_regGet(P,&reg);
        if (reg){
```

```
                mt_gotoXY(11,11*7);
                write(1,"P",1);
        }
        sc_regGet(O,&reg);
        if (reg){
                mt_gotoXY(11,11*7-2);
                write(1,"O",1);
        }
        sc_regGet(M,&reg);
        if (reg){
                mt_gotoXY(11,11*7+2);
                write(1,"M",1);
        }
        sc_regGet(T,&reg);
        if (reg){
                mt_gotoXY(11,11*7+4);
                write(1,"T",1);
        }
        sc_regGet(E,&reg);
        if (reg){
                mt_gotoXY(11,11*7+6);
                write(1,"E",1);
        }
        mt_gotoXY(13,8*7);
        write(1,"Keys",4);
        mt_gotoXY(14,8*7-2);
        write(1,"l - load",8);
        mt_gotoXY(15,8*7-2);
        write(1,"s - save",8);
        mt_gotoXY(16,8*7-2);
        write(1,"r - run",7);
        mt_gotoXY(17,8*7-2);
        write(1,"t - step",8);
        mt_gotoXY(18,8*7-2);
        write(1,"i - reset",9);
        mt_gotoXY(19,8*7-2);
        write(1,"F5 - accumulator",17);
        mt_gotoXY(20,8*7-2);
        write(1,"F6 - instructionCounter",23);
        char buf1[8];
        int l1 = sprintf(buf1,"+%.4x",value);
        int mass[8];
        for(i = 0; i < 5; i++){
                switch(buf1[i]){
                        case 'a': mass[i] = 10; break;
                        case 'b': mass[i] = 11; break;
                        case 'c': mass[i] = 12; break;
                        case 'd': mass[i] = 13; break;
                        case 'e': mass[i] = 14; break;
                        case 'f': mass[i] = 15; break;
                        case '+': mass[i] = 16; break;
                        default: mass[i] = buf1[i] - '0'; break;
                }
        }
        bc_printbigchar(big[mass[0]],14,2,BLACK,WHITE);
        bc_printbigchar(big[mass[1]],14,11,BLACK,WHITE);
        bc_printbigchar(big[mass[2]],14,20,BLACK,WHITE);
        bc_printbigchar(big[mass[3]],14,29,BLACK,WHITE);
        bc_printbigchar(big[mass[4]],14,38,BLACK,WHITE);
        mt_gotoXY(24,1);
}


void timer_on(int signo) {
        int t;
```

```
        sc_regGet(T,&t);
        if(t == 0){
                if(iC < 100){
                        iC++;
                        draw();
                        alarm(1);
                } else {
                        sc_regSet(T,1);
                        draw();
                        alarm(0);
                }
        } else {
                sc_regSet(T,1);
                draw();
                alarm(0);
        }
}

void timer_off(int signo) {
        sc_regSet(T,1);
        draw();
        alarm(0);
}

void move() {
        signal(SIGALRM, timer_on);
        signal(SIGUSR1, timer_off);
        draw();
}


int ALU(int command, int operand){
        switch(command){
                case 2*0x40: iC = operand; draw(); break;
                case 2*0x41: if(t1 < 0){ iC = operand; draw(); } break;
                case 2*0x42: if(t1 == 0){ iC = operand; draw(); } break;
                case 2*0x43: raise(SIGUSR1); break;
        }
}

int main(){
        struct itimerval nval, oval;
        nval.it_interval.tv_sec=2;
        nval.it_interval.tv_usec=500;
        nval.it_value.tv_sec=1;
        nval.it_value.tv_usec=0;
        sc_regInit();
        sc_regSet(T,1);
        mt_setfgcolor(WHITE);
        int r, c;
        mt_getscreensize(&r,&c);
        sc_memoryInit();
        rk_mytermregime(0,50,0,0,1);
        draw();
        enum keys key;
        while(1){
             rk_readkey(&key);
            int temp10;
            sc_regGet(T,&temp10);
            if(temp10 == 0){
               if(key == 5){
                        raise(SIGUSR1);
                        sc_memoryInit();
                        iC = 1;
                        draw();
```

```
   }
} else {
switch(key){
        case 2: sc_memorySave("lab5.memory"); break;
        case 3: {
           int temp;
           sc_regGet(T,&temp);
           if(temp){
                  sc_regSet(T,0);
                  setitimer(ITIMER_REAL, &nval, &oval);
                  move();
           } else {
                  sc_regSet(T,1);
                  alarm(0);
                  draw();
           }
        } break;
        case 1: sc_memoryLoad("lab5.memory"); draw(); break;
        case 9: if(iC > 10) iC -= 10; draw(); break;
        case 8: if(iC <= 90) iC += 10; draw(); break;
        case 10: if(iC > 1) iC--; draw(); break;
        case 11: if(iC < 100) iC++; draw(); break;
        case 12: {
           if(status == 1){
                  write(1,"Input: ",7);
                  rk_mytermregime(1,0,50,1,1);
                  read(1,&temp,4);
                  sscanf(temp,"%x",&t2);
                  status = 0;
                  mt_clrscr();
                  draw();
           } else if(status == 0){
                  if(verifMemory(t2) == 0)
                         sc_memorySet(iC,t2);
                  rk_mytermregime(0,50,0,0,1);
                  status = 1;
                  mt_clrscr();
                  draw();
           }
        } break;
        case 6: {
           if(statusAc == 1){
                  sc_memoryGet(iC,&t1);
                  statusAc = 0;
           } else if(statusAc == 0){
                  sc_memorySet(iC,t1);
                  statusAc = 1;
                  draw();
           }
        } break;
        case 7: {
           if(statusIc == 1){
                  write(1,"Input: ",7);
                  rk_mytermregime(1,0,50,1,1);
                  read(1,&temp1,4);
                  sscanf(temp1,"%x",&t2);
                  iC = t2;
                  mt_clrscr();
                  rk_mytermregime(0,50,0,0,1);
                  draw();
           }

        } break;
        case 5: {
           sc_memoryInit();
```

```
                    sc_regInit();
                    draw();
                } break;
                case 4: {
                    iC++;
                    draw();
                } break;
            }
        }
        }
        rk_mytermregime(1,0,50,1,1);
        return 0;
}
```