Министерство цифрового развития, связи и

массовых коммуникаций Российской Федерации

«СибГУТИ»

Кафедра ПМиК

Лабораторная работа №5

по

«Архитектуре ЭВМ»

Выполнил:

студент группы ИП-114

Амошенко А. И.

Проверил:

Мамойленко С. Н.

Новосибирск

2023

Ссылка на репозиторий: <https://git.csc.sibsutis.ru/ip114s01/mysimplecomputer>

В ветке lab5 были разработаны следующие файлы:

1. main.c - тестируется основная функциональность программы;

2. myTerm.h - объявляются все функции и константы;

3. myTerm.c - описываются все функции;

4 .gitignore игнорируются объектные файлы. В файле Makefile собирается основной проект;

5 .gitlab-ci.yml хранится код для сборки CI;

6. ALU – арифметико-логическое устройство

7. CU – управляющие устройство

8. Signal – отправляет сигналы;

9. myReadKey.c - статическая библиотека, описываются все функции;

10. myReadKey.h - объявляются все функции и макроссы;

11. makefile - собирается основной проект.

Список функций:

#ifndef MY\_ALU\_H #define MY\_ALU\_H

#include "mySimpleComputer.h"

int ALU (int command, int operand); #endif

#ifndef MY\_CU\_H #define MY\_CU\_H #include "myALU.h"

#include "myInterface.h" #include "myReadKey.h" #include "mySimpleComputer.h" #include "myTerm.h"

#include "string.h"

int READ (int operand); int WRITE (int operand); int LOAD (int operand); int STORE (int operand); int JUMP (int operand); int JNEG (int operand); int JZ (int operand); int HALT ();

int JNS (int operand); int CU ();

#endif

#ifndef MYSIGNAL\_H #define MYSIGNAL\_H

#include "myALU.h" #include "myCU.h" #include "myInterface.h"

#include "mySimpleComputer.h" #include <signal.h>

void ms\_setSignals ();

void ms\_timerHandler (int sig); void ms\_userSignal (int sig);

#endif // MYSIGNAL\_H

#ifndef MYBIGCHARS\_H #define MYBIGCHARS\_H

#include "myTerm.h" #include <stdbool.h> #include <stdio.h> #include <stdlib.h> #include <string.h> #include <unistd.h>

/\* Псевдографика \*/

#define ACS\_CKBOARD 'a' // Штриховка

#define ACS\_ULCORNER 'l' // Левый верхний угол #define ACS\_URCORNER 'k' // Правый верхний угол #define ACS\_LRCORNER 'j' // Правый нижний угол

#define ACS\_LLCORNER 'm' // Левый нижний угол #define ACS\_HLINE 'q' // Горизонтальная линия #define ACS\_VLINE 'x' // Вертикальная линия extern unsigned int

big\_chars[][2];

int bc\_printA (char charr); int bc\_box (int x1, int y1, int x2, int y2); int bc\_printBigChar (unsigned int \*big, int x, int y, enum colors colorFG, enum colors colorBG); int bc\_setBigCharPos (int \*big, int x, int y, int value); int bc\_getbigCharPos (int \*big, int x, int y, int \*value); int bc\_bigCharWrite (int fd, int \*big, int count); int bc\_bigCharRead (int fd, int \*big, int need\_count, int \*count); #endif

#ifndef INTERFACE\_H #define INTERFACE\_H #include "myBigChars.h"

#include "mySimpleComputer.h" #include <ctype.h>

#include <string.h> #include

<unistd.h> extern int instruction\_counter; int drawingBigChars (); int ui\_initial (); int ui\_update (); int ui\_setValue (); int drawingBoxes ();

int drawing\_texts (); int ui\_Counter (); int drawing\_memory (); int drawing\_flags (); int drawing\_IC (); int ui\_setValue (); bool checkCorrectInput (const char buffer[10]); int ui\_messageOutput (char \*str, enum colors color); int clearBuffIn ();

#endif

#ifndef MYREADKEY\_H #define MYREADKEY\_H

#include <stdbool.h> #include <stdio.h> #include <termios.h> #include <unistd.h>

enum keys

{

ESC\_KEY, L\_KEY, S\_KEY, R\_KEY, T\_KEY, I\_KEY, F5\_KEY, F6\_KEY, UP\_KEY, DOWN\_KEY, LEFT\_KEY, RIGHT\_KEY, ENTER\_KEY,

}; extern struct termios save;

int rk\_readKey (enum keys \*key); int rk\_myTermSave (); int rk\_myTermRestore (); int rk\_myTermRegime (int regime, int vtime, int vmin, int echo, int sigit);

#endif

#ifndef SIMPLECOMPUTER\_H #define SIMPLECOMPUTER\_H

#include <inttypes.h> #include <math.h> #include <stdio.h>

#include <unistd.h> // for write function #define N 100

#define REGISTER\_SIZE 5

#define OVERFLOW 0 // переполнение #define DIVISION\_ERROR\_BY\_ZERO 1 // ошибка деления на 0

#define OUT\_OF\_MEMORY 2 // выход за границы памяти

#define IGNORING\_CLOCK\_PULSES 3 // игнорирование тактовых импульсов #define INVALID\_COMMAND 4 // неверная команда

extern short int memory\_arr[N]; extern char flag;

extern short currMemCell;

int sc\_memoryInit (); int sc\_memorySet (int address, int value); int sc\_memoryGet (int address, int \*value); int sc\_memorySave (char \*filename); int sc\_memoryLoad (char

\*filename); int sc\_regInit (); int sc\_regSet (int \_register, int value); int sc\_regGet (int \_register, int \*value); int sc\_commandEncode (int command, int operand, int \*value); int sc\_commandDecode (int value, int \*command, int \*operand); #endif

#ifndef MYTERM\_H #define MYTERM\_H #include <stdio.h>

#include <sys/ioctl.h> #include

<unistd.h> // for write and dup2 enum

colors

{

RED = 196,

PEACH = 203,

GREEN = 10,

SOFT\_GREEN = 192,

BLUE = 20,

BLACK = 16,

GRAY = 240,

WHITE = 15,

DEFAULT = 0

};

int mt\_clrscr (); int mt\_gotoXY (int \_x, int

\_y); int mt\_getscreensize (int \*rows, int

\*cols); int mt\_setfgcolor (enum colors); int mt\_setbgcolor (enum colors); int set\_default\_color ();

#endif