PROGRAMMING PROJECT 2

LECTURER:ADEM ÖZYAVAŞ

ZEYNEP ECE TUZLA

220504025

#include <stdio.h>

#include <ctype.h>

#include <string.h>

#define **MAX\_LINE\_LENGTH 100 //max length of a line in the input file**

#define **REGISTER\_COUNT 32 //number of registers in the processor**

**int registers[REGISTER\_COUNT];//declares the registers**

// function is used to convert to string to integer

**int strToInt(const char \*str) {**

**int result = 0;**

**int i = 0;**

**while (str[i] != '\0' && isdigit(str[i])) {**

**result = result \* 10 + (str[i] - '0');**

**i++;**

**}**

**return result;**

**}**

//using for strstr function

//strstr function is used to find the substring in the string

**const char \*strFind(const char \*array, const char \*subarray) {**

**while (\*array) {**

**const char \*a = array;**

**const char \*s = subarray;**

**while (\*a == \*s && \*s) {**

**a++;**

**s++;**

**}**

**if (\*s == '\0') {**

**return array;** // substring is found

**}**

**array++;**

**}**

**return NULL; //** substring is not found

**}**

// function is used to process the put command

**void putCommand(const char \*line) {**

**int value = 0, reg = 0;**

**int i = 0;**

// searches the put command

**while (line[i]!='\0'&& !(line[i] == 'p' && line[i + 1] == 'u' && line[i + 2] == 't')) {**

**i++;**

**}**

// skip spaces

**while (line[i] == ' ') {**

**i++;**

**}**

// takes the value

**value = strToInt(line + i);**

// skip the spaces and the commas

**while (line[i] == ',') {**

**i++;**

**}**

// skips the character 'r'

**if (line[i] == 'r') {**

**i++;**

**}**

// takes the register number

**reg = line[i] - '0';**

// processes the put command

**if (reg >= 0 && reg < REGISTER\_COUNT) {**

**registers[reg] = value;**

**}**

**}**

// function is used to process the add command

**void addCommand(const char \*line) {**

**int reg1 = 0, reg2 = 0, destReg = 0;**

**int i = 0;**

// searches the add command

**while (line[i]!='\0'&& !(line[i] == 'a' && line[i + 1] == 'd' && line[i + 2] == 'd')) {**

**i++;**

**}**

// skip the spaces

**while (line[i] == ' ') {**

**i++;**

**}**

// skips the character 'r'

**if (line[i] == 'r') {**

**i++;**

**}**

// takes the first register number

**reg1 = line[i] - '0';**

//skip the commas

**while (line[i] == ',') {**

**i++;**

**}**

// skips the character 'r'

**if (line[i] == 'r') {**

**i++;**

**}**

// takes the second register number

**reg2 = line[i] - '0';**

//skip the commas

**while(line[i] == ','){**

**i++;**

**}**

//skips the character 'r'

**if(line[i]=='r'){**

**i++;**

**}**

// takes the destination register number

**destReg = line[i] - '0';**

// processes the add command

**if (reg1 >= 0 && reg1 < REGISTER\_COUNT && reg2 >= 0 && reg2 < REGISTER\_COUNT && destReg >= 0 && destReg < REGISTER\_COUNT) {**

**registers[destReg] = registers[reg1] + registers[reg2];**

**}**

**}**

// function is used to process the prn command

**void prnCommand(const char \*line) {**

**int reg = 0;**

**int i = 0;**

// searches the prn command

**while (line[i]!='\0'&& !(line[i] == 'p' && line[i + 1] == 'r' && line[i + 2] == 'n')) {**

**i++;**

**}**

// skip the spaces

**while (line[i] == ' ') {**

**i++;**

**}**

// skips the character 'r'

**if (line[i] == 'r') {**

**i++;**

**}**

// takes the register number

**reg = line[i] - '0';**

// processes the prn command

**if (reg >= 0 && reg < REGISTER\_COUNT) {**

**printf("%d\n", registers[reg]);**

**}**

**}**

// function is used to process the jmpe command

**void jmpeCommand(const char \*line, int \*currentLine) {**

**int reg1 = 0, reg2 = 0, destLine = 0;**

**int i = 0;**

// searches the jmpe command

**while (line[i]!='\0'&& !(line[i] == 'j' && line[i + 1] == 'm' && line[i + 2] == 'p' && line[i + 3] == 'e')) {**

**i++;**

**}**

// skip the spaces

**while (line[i] == ' ') {**

**i++;**

**}**

//skips the character 'r'

**if (line[i] == 'r') {**

**i++;**

**}**

// takes the first register number

**reg1 = line[i] - '0';**

//skip the commas

**while (line[i] == ',') {**

**i++;**

**}**

//skips the character 'r'

**if (line[i] == 'r') {**

**i++;**

**}**

// takes the second register number

**reg2 = line[i] - '0';**

// skip the commas

**while(line[i] == ',') {**

**i++;**

**}**

// takes the destination register number

**destLine = strToInt(line+i);**

// processes the jmpe command

**if (reg1 >= 0 && reg1 < REGISTER\_COUNT && reg2 >= 0 && reg2 < REGISTER\_COUNT) {**

**if (registers[reg1] == registers[reg2]) {**

**\*currentLine = destLine-1;** // -1 because it is incrementing in the loop

**}**

**}**

**}**

// function is used to process the jmpu command

**void jmpuCommand(const char \*line, int \*currentLine) {**

**int destLine = 0;**

**int i = 0;**

// searches the jmpu command

**while (line[i]!='\0'&& !(line[i] == 'j' && line[i + 1] == 'm' && line[i + 2] == 'p' && line[i + 3] == 'u')) {**

**i++;**

**}**

// skip the spaces

**while (line[i] == ' ') {**

**i++;**

**}**

// takes the destination register number

**destLine = strToInt(line + i);**

// processes the jmpu command

**\*currentLine = destLine-1 ; // -1 because it is incrementing in the loop**

**}**

// function is used to process the halt command

**void haltCommand(int \*running) {**

// processes the halt command

**\*running = 0;**

**}**

//this loop checks if the line contains any of the commands in the SimLan

**void processCommand(const char \*line, int \*running, int \*currentLine) {**

**if (strFind(line, "put") != NULL) {**

**putCommand(line);**

**} else if (strFind(line, "add") != NULL) {**

**addCommand(line);**

**} else if (strFind(line, "prn") != NULL) {**

**prnCommand(line);**

**} else if (strFind(line, "jmpe") != NULL) {**

**jmpeCommand(line, currentLine);**

**} else if (strFind(line, "jmpu") != NULL) {**

**jmpuCommand(line, currentLine);**

**} else if (strFind(line, "halt") != NULL) {**

**haltCommand(running);**

**\*running = 0;**

**}**

**}**

**int main() {**

**FILE \*file=fopen("program2.sla","r");//**opens the file

**char line[MAX\_LINE\_LENGTH];//**declares the line

**int currentLine=0;//**declares and initializes the current line

**int running=1;** //declares and initializes the running variable

// this loop reads the file line by line

**while (fgets(line, MAX\_LINE\_LENGTH, file) != NULL && running) {**

**processCommand(line, &running, &currentLine);** //calls the process command function

**printf("%s", line);**

**currentLine++;**

**}**

// closes the file

**fclose(file);**

**return 0;**

**}**

//but this program just prints the commands in the file,it couldn't print the sum of the numbers and the factorial of the numbers because there is something missing probably and I never found it no matter how hard I tried and I made my best about that but I couldn't find it.

**MY OUTPUT**

**put 0,r1**

**put 1,r2**

**put 10,r3**

**put 1,r4**

**jmpe r2,r3,9**

**add r2,r1**

**add r4,r2**

**jmpu 5**

**prn r1**

**halt**