Omar Garcia

Programing Languages

Project 2

Functional Decomposition

User-defined data structures used as parameters in the functions

Files and Functions

/\*Main file focus on setting up the hash table, opening the file, and starting the parser \*/

Main.c

/\*Initilizes all prerequisites

Sets the first two values of the hash table to begin and end

The program then tries to open all file passed onto it\*/

int main(int argc, char \*argv[]);

/\*This method opens the file. If the file doeas not exist, an error is thrown

Once the file is opened, the program checks for the begin key word to start the program. The program is then passed onto the begin function\*/

void openF(char \*str);

/\*This function sets lookahead to first identifier and launches the parser\*/

void Begin(char \*str);

/\*Parser uses look ahead in the program and uses a recursive algorithm to determine whether a statement is valid\*/

Parse.c

/\*This function determines whether the next value is legal\*/

void match(int t);

/\*This function starts the assignment of an identifier\*/

void AssignStmt();

/\*This function tests to see if any expressioins are present\*/

void Expression();

/\*This function tries to find if the assignment has any terms\*/

void Term();

/\*Checks the factors, whether it is an id, number, or ()\*/

void Factor();

/\*Lexan finds what the next value is. It determines whether it is begin, end , an id, or a number\*/

Lexan.c

/\*Lexan is used to determine what the next element will be\*/

int lexan();

/\*Function to lookup the position of an identifier\*/

int lookup(char \* c);

/\*This source file handles the errors that may occur in the program. \*/

ErrorHandler.c

/\*Simple method that handles error. Assigns a message that is appropriate for the error\*/

void errorHandler(int t, int lookahead)

/\*Handles the overall errors of the program. Prints out the error for the user\*/

void programErrors(int lookahead, char\* str)

/\*Reads the entire file and prints it for the user\*/

void readFile(char\* str)

/\*Creates a string to display information to the user.\*/

Stringer.c

/\*Concatinates the information needed to be displayed\*/

void add(int x, char\* program)

/\*Prints out a message to tell the user it is compiling. It displays the informtion of variables being stored into registers. It also prints out every vatraible us at the end\*/

void toString(char\* proName, char\* program, char\* hashTable[])