|  |
| --- |
|  |
| Data Dictionary for SimpleCompiler.h |
|  |
|  |
| **Andrew Laing** |
| **24/10/2017** |

The Simple Compiler

The Simple Programming language is, as its name implies, a very simple programming language with a limited set of instructions. It is a subset of the Basic language, lacking a lot of Basic’s features.

This Simple Compiler creates files of SML instructions from Simple source files. The compiler runs two passes on the Simple source code. On the first pass it constructs a lookup table from each line and generates most of the instructions. Goto addresses which it could not find in the lookup table during the first pass are resolved during the second pass, and the instruction are then written out to ‘out.sml’

|  |  |  |
| --- | --- | --- |
| **Defines** | | |
| **Name** | **Value** | **Description** |
| ADDINSTRUCTION(instruction) |  | A Macro equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* Used to improve code readability. |
| MEMORYSIZE | 1000 | Used to define the size of the SML Virtual Memory space. |
| MAXLINELENGTH | 100 | Used to limit the length of lines which can be read in from a Simple source file. |
| MAXTOKENLENGTH | 20 | Used to define the maximum length of tokens, extracted from a line of Simple source code. Note: Simple variables are one character long and comments are skipped, so the only other things being tokenised are keywords, operators and line numbers. |
| MAXTABLESIZE | 100 | Used to define the size of the Symbol Table. |
| NUMBEROFCOMPOPERATORS | 6 | Used to define the maximum number of comparison operators allowed within the Simple language. |
| MAXARRAYSIZE | 20 | Used to define the maximum allowed size for a Simple array. (Introduced because of the restricted size of the MEMORY size which could result in array elements overwriting instructions.) |
| SUCCESS | 1 | Used as a return value for a successful operation to make the code more readable. |
| FAIL | -1 | Used as a return value for a failed operation to make the code more readable. |

|  |  |  |
| --- | --- | --- |
| **Global Variables** | | |
| **Name** | **Type** | **Description** |
| tableEntry | struct | Used to hold single entries within the Symbol Table. It holds the three values which follow. |
| tableEntry.symbol | int | Used to hold the symbol encountered whilst running the first pass through the source code. This can be either a line number, a variable, or a constant. |
| tableEntry.type | char | Used to hold a character showing which type of symbol tableEntry.symbol holds. This can be either ‘L’ (line number), ‘V’ (variable), or ‘C’ (constant). |
| tableEntry.location | int | Used to hold the memory address where the value attached to the symbol will be stored, or the instructions created from the line of Simple source code will start from. |
| Table | struct tableEntry [] | An array of tableEntries. I.e., the Symbol table. |
| ComparisonOperators | enum | Used to make the code easier to read, their values correlate to the indicies of the comparison operators they refer to in the compOperators array. |
| compOperators | Const char [][] | Used to hold string representations of Simple’s comparison operators which are used in creating instructions during the compiler’s first pass. |
| ForLoopStackPtr | ForLoopNodePtr | Used to hold a pointer to the top of the stack. ForLoopNodePtr is defined in ForLoopStack.h |
| stringLiteralArray | char[][] | Used to hold string literals entered within SIMPL source code so that they can be reused. |
| stringLiteralAddresses | int[] | Used to hold the MEMORY addresses where the literals within the stringLiteralArray are stored. |
| numberOfStringLiterals | int | Used to hold the number of string literals added to MEMORY. |
| TABLEPOSITION | int | Used to hold the next available position in the Symbol Table which can accept an entry. |
| Flags | int [] | Used to hold the SML addresses of instructions which need their operands, eg BRANCH addresses, resolved during the second pass. |
| PREVIOUSLINENUMBER | int | Used to hold the line number of the previously read Simple line to ensure that they have been numbered correctly. (Incorrect ordering will result in the program terminating with a Syntax Error) |
| INSTRUCTIONCOUNTER | int | Used to hold the address with the SML Memory space where the next instruction can be added. |
| VARCONSTADDRESS | int | Used to hold the next memory address, at the end of the SML Memory space, where a value can be added |
| MEMORY | Int array | Used to store all of the instructions, variables and constants necessary to running an SML program. |