|  |
| --- |
|  |
| Data Dictionary for  main.c |
|  |
|  |
| **Andrew Laing** |
| **24/10/2017** |

Contents

[Using the Simple Compiler 3](#_Toc496557736)

[main () 3](#_Toc496557737)

[initialiseFlags() 3](#_Toc496557738)

[printFlags() 3](#_Toc496557739)

[printInstructions() 4](#_Toc496557740)

[addInstructionToMemory() 4](#_Toc496557741)

[writeInstructionsToFile() 4](#_Toc496557742)

[addLineNumberToTable() 5](#_Toc496557743)

[updateStringVariableAddress() 5](#_Toc496557744)

[addVariableToTable () 6](#_Toc496557745)

[addConstantToTable() 6](#_Toc496557746)

[runFirstPass() 7](#_Toc496557747)

[runSecondPass() 7](#_Toc496557748)

[createTokenArray() 8](#_Toc496557749)

[addSymbolsToTable() 8](#_Toc496557750)

[getElementAddressFromStr() 9](#_Toc496557751)

[createInputInstruction() 9](#_Toc496557752)

[addStringLiteralToMemory() 10](#_Toc496557753)

[createPrintInstruction() 10](#_Toc496557754)

[createGotoInstruction() 11](#_Toc496557755)

[createEndInstruction() 12](#_Toc496557756)

[addStringToVariableMemorySpace() 12](#_Toc496557757)

[createLetInstruction() 13](#_Toc496557758)

[createDimInstruction() 14](#_Toc496557759)

[createIfInstruction() 14](#_Toc496557760)

[createGosubInstruction() 15](#_Toc496557761)

[createReturnInstruction() 16](#_Toc496557762)

[findStepValueIndex() 17](#_Toc496557763)

[findToValueIndex() 17](#_Toc496557764)

[createForInstruction() 17](#_Toc496557765)

[createNextInstruction() 18](#_Toc496557766)

[createNewlineInstruction() 19](#_Toc496557767)

[dumpTableEntries() 19](#_Toc496557768)

[variableExistsInTable() 19](#_Toc496557769)

[constantExistsInTable() 20](#_Toc496557770)

[lineNumberExistsInTable() 20](#_Toc496557771)

[convertToPostfix () 20](#_Toc496557772)

[precedence() 20](#_Toc496557773)

[evaluatePostfixExpression() 21](#_Toc496557774)

[calculate() 21](#_Toc496557775)

[getComparisonOperatorIndex() 22](#_Toc496557776)

[isComparisonOperator() 23](#_Toc496557777)

[doLTJMP() 23](#_Toc496557778)

[doGTJMP() 25](#_Toc496557779)

[doLTEJMP() 26](#_Toc496557780)

[doGTEJMP() 28](#_Toc496557781)

[doEQJMP() 29](#_Toc496557782)

[doNEJMP() 31](#_Toc496557783)

[doSTRCMP() 32](#_Toc496557784)

[doSTRCMP() 33](#_Toc496557785)

# Using the Simple Compiler

To use the Simple Compiler just run it from the command line. It will ask for the name of the file to compile, which should have a ‘.simple ’ suffix. This Simple file should be located within the same directory as the compiler binary. The compiled SML file will also be written out to this directory as ‘out.sml’

|  |  |  |
| --- | --- | --- |
| **main ()** | | |
| **Name** | **Type** | **Description** |
| filename | char [] | Used to hold the name of the Simple source code entered by the user. |
| MAXLINELENGTH | 100 | Used to limit the length of the filename read in. Defined in SimpleCompiler.h |
| SUCCESS | 1 | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h |
| FAIL | -1 | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h |

|  |  |  |
| --- | --- | --- |
| **initialiseFlags()** | | |
| **Name** | **Type** | **Description** |
| i | int | Used to iterate through the Flags array. |
| MEMORYSIZE |  | Used to define the size of the SML Virtual Memory space. |
| Flags | int [] | Used to hold the SML addresses of instructions which need their operands, eg BRANCH addresses, resolved during the second pass. Defined in SimpleCompiler.h |

|  |  |  |
| --- | --- | --- |
| **printFlags()** | | |
| **Name** | **Type** | **Description** |
| i | int | Used to iterate through the Flags array. |
| MEMORYSIZE |  | Used to define the size of the SML Virtual Memory space. |
| Flags | int [] | Used to hold the SML addresses of instructions which need their operands, eg BRANCH addresses, resolved during the second pass. Defined in SimpleCompiler.h |

|  |  |  |
| --- | --- | --- |
| **printInstructions()** | | |
| **Name** | **Type** | **Description** |
| i | int | Used to iterate through the SML Virtual Memory space array. |
| MEMORYSIZE |  | Used to hold the size of the SML Virtual Memory space. Defined in SimpleCompiler.h |
| MEMORY | Int array | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |

|  |  |  |
| --- | --- | --- |
| **addInstructionToMemory()** | | |
| **Name** | **Type** | **Description** |
| INSTRUCTIONCOUNTER | int | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| VARCONSTADDRESS | int | Used to hold the next memory address, at the end of the SML Memory space, where a value can be added. Defined in SimpleCompiler.h |
| MEMORY | Int array | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| SUCCESS | 1 | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h |
| FAIL | -1 | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h |

|  |  |  |
| --- | --- | --- |
| **writeInstructionsToFile()** | | |
| **Name** | **Type** | **Description** |
| i | int | Used to iterate through the SML Virtual Memory space array. |
| ofPtr | FILE \* | Used to hold a pointer to the output file containing the SML instructions. |
| MEMORYSIZE |  | Used to hold the size of the SML Virtual Memory space. Defined in SimpleCompiler.h |
| MEMORY | Int array | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| SUCCESS | 1 | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h |
| FAIL | -1 | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h |

|  |  |  |
| --- | --- | --- |
| **addLineNumberToTable()** | | |
| **Name** | **Type** | **Description** |
| Table | struct tableEntry [] | An array of tableEntries. I.e., the Symbol table. Defined in SimpleCompiler.h |
| TABLEPOSITION | int | Used to hold the next available position in the Symbol Table which can accept an entry. Defined in SimpleCompiler.h |
| INSTRUCTIONCOUNTER | int | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| 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.) Defined in SimpleCompiler.h |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **updateStringVariableAddress()** | | | | |
| **Name** | **Type** | | **Description** | |
| i | int | | Used to iterate through the symbol Table looking for the entry relating to the string variable. | |
| entryUpdated | int | | Used as a flag to indicate that variable entry was updated correctly and is the return value of the function. | |
| offset | int | | Used in calculating the new memory address for the string variable. | |
| newPosition | int | | Used to hold the new MEMORY location for the stringf variable. | |
| Table | struct tableEntry [] | | An array of tableEntries. I.e., the Symbol table. Defined in SimpleCompiler.h | |
| TABLEPOSITION | int | | Used to hold the next available position in the Symbol Table which can accept an entry. Defined in SimpleCompiler.h | |
| INSTRUCTIONCOUNTER | | int | | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| MEMORY | Int array | | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h | |
| VARCONSTADDRESS | int | | Used to hold the next memory address, at the end of the SML Memory space, where a value can be added. Defined in SimpleCompiler.h | |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |
| --- | --- | --- |
| **addVariableToTable ()** | | |
| **Name** | **Type** | **Description** |
| Table | struct tableEntry [] | An array of tableEntries. I.e., the Symbol table. Defined in SimpleCompiler.h |
| TABLEPOSITION | int | Used to hold the next available position in the Symbol Table which can accept an entry. Defined in SimpleCompiler.h |
| VARCONSTADDRESS | int | Used to hold the next memory address, at the end of the SML Memory space, where a value can be added. Defined in SimpleCompiler.h |
| SUCCESS | 1 | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h |
| FAIL | -1 | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h |

|  |  |  |
| --- | --- | --- |
| **addConstantToTable()** | | |
| **Name** | **Type** | **Description** |
| Table | struct tableEntry [] | An array of tableEntries. I.e., the Symbol table. Defined in SimpleCompiler.h |
| TABLEPOSITION | int | Used to hold the next available position in the Symbol Table which can accept an entry. Defined in SimpleCompiler.h |
| MEMORY | Int array | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| VARCONSTADDRESS | int | Used to hold the next memory address, at the end of the SML Memory space, where a value can be added. Defined in SimpleCompiler.h |
| SUCCESS | 1 | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h |
| FAIL | -1 | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h |

|  |  |  |
| --- | --- | --- |
| **runFirstPass()** | | |
| **Name** | **Type** | **Description** |
| line | char [] | Used to hold a line of code read in from the Simple source file |
| lineCopy | char [] | Used to hold a copy of line because when the line is tokenised into the tokenArray using strtok, this changes the original string. |
| tokenArray | char [][] | Used to hold the tokens created from the line which will be added to the Symbol Table and used to create instructions. |
| lineParsed | int | Used as a flag to determine whether lines of source code have been parsed correctly. |
| sourceFilePtr | FILE \* | Used to hold a pointer to the Simple source file being compiled into SML. |
| numberOfTokens | int | Used to hold the number of tokens created from the line of Simple source code, and added to tokenArray. |
| SUCCESS | 1 | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h |
| FAIL | -1 | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h |

|  |  |  |
| --- | --- | --- |
| **runSecondPass()** | | |
| **Name** | **Type** | **Description** |
| i | int | Used to iterate through the Flags array and update the SML MEMORY address with the resolved line number address for a Branch instruction. |
| address | int | Used to hold resolved address to add to the SML instruction which did not resolve it during the first pass of the compiler. |
| MEMORYSIZE |  | Used to hold the size of the SML Virtual Memory space. Defined in SimpleCompiler.h |
| MEMORY | Int array | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| Flags | int [] | Used to hold the SML addresses of instructions which need their operands, eg BRANCH addresses, resolved during the second pass. Defined in SimpleCompiler.h |
| SUCCESS | 1 | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h |
| FAIL | -1 | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h |

|  |  |  |
| --- | --- | --- |
| **createTokenArray()** | | |
| **Name** | **Type** | **Description** |
| length | int | Used to hold the number of tokens added to the tokenArray, and is the return value of this function. |
| lineCopy | char [] | Used to hold a copy of the line passed to this function because passing the original to strtok would alter it. |
| tokenPtr | char \* | Used to hold a reference to tokens created by strtok. |
| MAXLINELENGTH |  | Used to limit the length of lines which can be read in from a Simple source file. Defined in SimpleCompiler.h |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **addSymbolsToTable()** | | | | |
| **Name** | | **Type** | | **Description** |
| number | | int | | Used to hold numbers, extracted from the Simple source code, to be added to the Symbol Table. |
| isLineNumber | | int | | Used to as a flag to process line numbers extracted from the Simple source code, to be added to the Symbol Table. |
| i | | int | | Used to iterate through the token array. |
| 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.) Defined in SimpleCompiler.h |
| INSTRUCTIONCOUNTER | | int | | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **getElementAddressFromStr()** | | | | |
| **Name** | | **Type** | | **Description** |
| arraySize | | int | | Used to hold the size of the memory space allocated to the array. |
| multiplier | | int | | Used to convert the array index stored as characters into an integer. |
| arrayIdx | | int | | Used to iterate through the token. |
| idx | | int | | Used to hold the index of the element array to assign the variable to. |
| MEMORY | | Int array | | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **createInputInstruction()** | | | | |
| **Name** | | **Type** | | **Description** |
| instruction | | int | | Used to hold the instruction to be written to the SML Memory space. |
| variableAddress | | int | | Used to hold the address within the SML Memory space containing the value for the variable to be READ. |
| lineNumber | | int | | Used to hold the line number attached to the Simple Command, converted to an integer. |
| instructionAddress | | int | | Used to hold the address assigned to the line number within the SML Memory space where its first instruction will be stored. |
| ADDINSTRUCTION | |  | | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| MEMORY | | Int array | | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| READ | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **addStringLiteralToMemory()** | | | | |
| **Name** | | **Type** | | **Description** |
| copy | | char[] | | Used to hold a copy of the string literal to be added to MEMORY. |
| MAXLINELENGTH | |  | | Used to limit the length of lines which can be read in from a Simple source file. Defined in SimpleCompiler.h |
| idx | | int | | Used to iterate through the tokenArray whilst adding the tokens to copy. |
| literalAddress | | int | | Used to hold the address assigned to the string literal within the SML Memory space where its first instruction will be stored. |
| stringLen | | int | | Used to hold the string length of the string literal to be stored in MEMORY. |
| offset | | int | | Used in calculating the number of MEMORY addresses to be assigned to the string literal. |
| startPosition | | int | | Used to hold the address of the string literal in MEMORY. |
| VARCONSTADDRESS | | int | | Used to hold the next memory address, at the end of the SML Memory space, where a value can be added. Defined in SimpleCompiler.h |
| INSTRUCTIONCOUNTER | | int | | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| address | | int | | Used when adding the string literal to MEMORY. |
| i | | int | | Used to iterate through the string literal. |
| temp | | int | | Used to temporarily store characters to be written to memory from the string literal. |
| MEMORY | | Int array | | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **createPrintInstruction()** | | | | |
| **Name** | | **Type** | | **Description** |
| instruction | | int | | Used to hold the instruction to be written to the SML Memory space. |
| variableAddress | | int | | Used to hold the address within the SML Memory space containing the value for the variable to be written to STDOUT. |
| variableIdx | | int | | Used to hold the index of variables to be printed within the tokenArray. |
| ADDINSTRUCTION | |  | | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| MEMORY | | Int array | | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| WRITE | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| WRITESTRING | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **createGotoInstruction()** | | | | |
| **Name** | | **Type** | | **Description** |
| instruction | | int | | Used to hold the instruction to be written to the SML Memory space.. |
| returnLineNumber | | int | | Used to hold the line number of Simple code representing the line where the goto statement will jump to. |
| returnAddress | | int | | Used to hold the address assigned to the return line number address within the SML Memory space where program execution will branch to. |
| ADDINSTRUCTION | |  | | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| MEMORY | | Int array | | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| Flags | | int [] | | Used to hold the SML addresses of instructions which need their operands, eg BRANCH addresses, resolved during the second pass. Defined in SimpleCompiler.h |
| INSTRUCTIONCOUNTER | | int | | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| BRANCH | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **createEndInstruction()** | | | | |
| **Name** | | **Type** | | **Description** |
| instruction | | int | | Used to the instruction to be written to the SML Memory space. |
| ADDINSTRUCTION | |  | | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| HALT | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **addStringToVariableMemorySpace()** | | | | |
| **Name** | | **Type** | | **Description** |
| copy | | char[] | | Used to hold a copy of the string literal to be added to MEMORY. |
| MAXLINELENGTH | |  | | Used to limit the length of lines which can be read in from a Simple source file. Defined in SimpleCompiler.h |
| idx | | int | | Used to iterate through the tokenArray whilst adding the tokens to copy. |
| address | | int | | Used to hold the address assigned to the string literal within the SML Memory space where its first instruction will be stored. |
| stringLen | | int | | Used to hold the string length of the string literal to be stored in MEMORY. |
| INSTRUCTIONCOUNTER | | int | | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| address | | int | | Used when adding the string literal to MEMORY. |
| i | | int | | Used to iterate through the string literal. |
| temp | | int | | Used to temporarily store characters to be written to memory from the string literal. |
| MEMORY | | Int array | | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **createLetInstruction()** | | | | |
| **Name** | | **Type** | | **Description** |
| INSTRUCTIONCOUNTER | | int | | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| postfix | | char [][] | | Used to hold the expression on the right-hand side of the assignment operator in the Simple command, converted from infix to postfix. |
| pfLength | | int | | Used to hold the number of tokens added to the postfix array. |
| variableAddress | | int | | Used to hold the address within the SML Memory space containing where the result of the postfix evaluation will be stored. |
| loadInstruction | | int | | Used to hold the LOAD instruction to be written to the SML Memory space. |
| storeInstruction | | int | | Used to hold the STORE instruction to be written to the SML Memory space. |
| evalResult | | int | | Used to hold the temporary address in the SML Memory space where the result of evaluation the expression on the right-hand side of the assignment operator will be stored. |
| ADDINSTRUCTION | |  | | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| MEMORY | | Int array | | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| BRANCH | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| LOAD | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| STORE | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **createDimInstruction()** | | | | |
| **Name** | | **Type** | | **Description** |
| INSTRUCTIONCOUNTER | | int | | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| multiplier | | int | | Used to convert the array size stored as characters into an integer. |
| arraySize | | int | | Used to hold the size of the array; this is stored as the first item in the array to indicate how many memory addresses have been reserved for its elements. |
| idx | | int | | Used to iterate through the token array when converting the arraySize to an integer. |
| addressOfVariable | | int | | Used to hold the first address assigned to the array variable in memory. This is where its arraySize will be stored. |
| i | | int | | Used when reserving memory space for the array elements. |
| VARCONSTADDRESS | | int | | Used to hold the next memory address, at the end of the SML Memory space, where a value can be added. Defined in SimpleCompiler.h |
| MEMORY | | Int array | | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **createIfInstruction()** | | | | |
| **Name** | | **Type** | | **Description** |
| INSTRUCTIONCOUNTER | | int | | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| compOpIdx | | int | | Used to hold the index of the comparison operator within the token array. Will be used to split the array into two expressions to evaluate then compare. |
| ADDINSTRUCTION | |  | | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| LTOPERATOR | |  | | An item from the ComparisonOperators enum. |
| GTOPERATOR | |  | | An item from the ComparisonOperators enum. |
| LTEOPERATOR | |  | | An item from the ComparisonOperators enum. |
| GTEOPERATOR | |  | | An item from the ComparisonOperators enum. |
| EQOPERATOR | |  | | An item from the ComparisonOperators enum. |
| NEOPERATOR | |  | | An item from the ComparisonOperators enum. |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **createGosubInstruction()** | | | | |
| **Name** | | **Type** | | **Description** |
| INSTRUCTIONCOUNTER | | int | | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| returnAddress | | int | | Used to hold the address that the program will branch back to after running the subroutine called by the GOSUB statement. |
| ADDINSTRUCTION | |  | | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| loadInstruction | | int | | Used to hold the LOAD instruction to be written to the SML Memory space. |
| storeInstruction | | int | | Used to hold the STORE instruction to be written to the SML Memory space. |
| gotoInstruction | | int | | Used to hold the BRANCH instruction to be written to the SML Memory space. |
| gosubVarAddress | | int | | Used told the address where the return address will be stored in the SML Memory space. |
| gotoLineNumber | | int | | Used to hold the line number of the subroutine to branch to. |
| gotoAddress | | int | | Used to hold the address in the SML memory space where the first instruction of the subroutine starts. |
| MEMORY | | Int array | | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| BRANCH | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| LOAD | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| STORE | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **createReturnInstruction()** | | | | |
| **Name** | | **Type** | | **Description** |
| INSTRUCTIONCOUNTER | | int | | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| Instruction1 | | int | | Used to hold the LOAD instruction to be written to the SML Memory space. |
| ADDINSTRUCTION | |  | | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| addInstruction | | int | | Used to hold the ADD instruction to be written to the SML Memory space. |
| storeInstruction | | int | | Used to hold the STORE instruction to be written to the SML Memory space. |
| returnAddress | | int | | Used to hold the address that the program will branch back to after running the subroutine called by the GOSUB statement. |
| MEMORY | | Int array | | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| BRANCH | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| ADD | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| LOAD | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| STORE | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **findStepValueIndex()** | | | | |
| **Name** | | **Type** | | **Description** |
| svi | | int | | Used to hold the index of the keyword STEP in the tokenArray and is the return value. Returned as -1 if the keyword is not found. |
| idx | | int | | Used to iterate through the tokenArray. |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **findToValueIndex()** | | | | |
| **Name** | | **Type** | | **Description** |
| tvi | | int | | Used to hold the index of the keyword TO in the tokenArray and is the return value. Returned as -1 if the keyword is not found. |
| idx | | int | | Used to iterate through the tokenArray. |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |  |
| --- | --- | --- | --- |
| **createForInstruction()** | | | |
| **Name** | **Type** | | **Description** |
| INSTRUCTIONCOUNTER | int | | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| ADDINSTRUCTION |  | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* | |
| addressOfCountVariable | int | | Used to hold the address in the MEMORY space of the count variable. This is added to the ForLoopStack to be used by the createNextInstruction function. |
| addressOfStepValue | int | | Used to hold the address in the MEMORY space of the step value. This is added to the ForLoopStack to be used by the createNextInstruction function. |
| startOfLoopAddress | int | | Used to hold instruction number at the start of the for loop. This is added to the ForLoopStack to be used by the createNextInstruction function. |
| stepIndex | int | | Used to hold the index of the STEP keyword in the token array. |
| stepIndex | int | | Used to hold the index of the TO keyword in the token array. |
| letStatement | char [][] | | Used to create a token array to send to the createLetInstruction function |
| jmpGTEStatement | char [][] | | Used to create a token array to send to the doJMPGT function. |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **createNextInstruction()** | | | | |
| **Name** | | **Type** | | **Description** |
| INSTRUCTIONCOUNTER | | int | | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| stackPopped | | int[] | | Used to hold the popped values from the ForLoopStack. |
| ADDINSTRUCTION | |  | | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| loadInstruction | | int | | Used to hold the LOAD instruction to be written to the SML Memory space. |
| addInstruction | | int | | Used to hold the ADD instruction to be written to the SML Memory space. |
| storeInstruction | | int | | Used to hold the STORE instruction to be written to the SML Memory space. |
| MEMORY | | Int array | | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| BRANCH | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| ADD | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| LOAD | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| STORE | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| branchPosInstruction | | int | | Used to hold the BRANCHPOS instruction to be written to the SML Memory space. |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **createNewlineInstruction()** | | | | |
| **Name** | | **Type** | | **Description** |
| instruction | | int | | Used to the instruction to be written to the SML Memory space. |
| ADDINSTRUCTION | |  | | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| NEWLINE | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |
| --- | --- | --- |
| **dumpTableEntries()** | | |
| **Name** | **Type** | **Description** |
| Table | struct tableEntry [] | An array of tableEntries. I.e., the Symbol table. Defined in SimpleCompiler.h |
| TABLEPOSITION | int | Used to hold the next available position in the Symbol Table which can accept an entry. Defined in SimpleCompiler.h |
| i | int | Used to iterate through the Symbol table. |

|  |  |  |
| --- | --- | --- |
| **variableExistsInTable()** | | |
| **Name** | **Type** | **Description** |
| Table | struct tableEntry [] | An array of tableEntries. I.e., the Symbol table. Defined in SimpleCompiler.h |
| TABLEPOSITION | int | Used to hold the next available position in the Symbol Table which can accept an entry. Defined in SimpleCompiler.h |
| i | int | Used to iterate through the Symbol table. |

|  |  |  |
| --- | --- | --- |
| **constantExistsInTable()** | | |
| **Name** | **Type** | **Description** |
| Table | struct tableEntry [] | An array of tableEntries. I.e., the Symbol table. Defined in SimpleCompiler.h |
| TABLEPOSITION | int | Used to hold the next available position in the Symbol Table which can accept an entry. Defined in SimpleCompiler.h |
| i | int | Used to iterate through the Symbol table. |

|  |  |  |
| --- | --- | --- |
| **lineNumberExistsInTable()** | | |
| **Name** | **Type** | **Description** |
| Table | struct tableEntry [] | An array of tableEntries. I.e., the Symbol table. Defined in SimpleCompiler.h |
| TABLEPOSITION | int | Used to hold the next available position in the Symbol Table which can accept an entry. Defined in SimpleCompiler.h |
| i | int | Used to iterate through the Symbol table. |

|  |  |  |
| --- | --- | --- |
| **convertToPostfix ()** | | |
| **Name** | **Type** | **Description** |
| i | int | Used to iterate through the infix array. |
| postFixLength | int | Used to add characters to the postfix array |
| c | char | Used to hold characters from the infix array to be pushed to the stack |
| stackPtr | StackNodePtr | Used to create the postfix expression. Defined in IntStack.h |

|  |  |  |
| --- | --- | --- |
| **precedence()** | | |
| **Name** | **Type** | **Description** |
| result | int | Used to hold the result of the comparison operation which determines the operator with the higher precedence, and is the retrurn value of this function. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **evaluatePostfixExpression()** | | | | |
| **Name** | | **Type** | | **Description** |
| stackPtr | | StackNodePtr | | Used when evaluating the postfix expression. Defined in IntStack.h |
| x | | int | | Used to hold the address of a variable/constant SML Memory space address popped from the stack to create an instruction with. |
| y | | int | | Used to hold a variable/constant SML Memory space address popped from the stack to create an instruction with. |
| i | | int | | Used to iterate through the postfix array. |
| address | | int | | Used to hold a variable/constant SML Memory space address to push to the stack. |
| c | | char | | Used to determine what the token in the postfix array represents, eg variable, operator… |
| number | | int | | Used when converting tokens to integers. |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **calculate()** | | | | |
| **Name** | | **Type** | | **Description** |
| tempStorage | | int | | Used to hold an address in the SML Memory space where the result of a calculation will be stored. |
| VARCONSTADDRESS | | int | | Used to hold the next memory address, at the end of the SML Memory space, where a value can be added. Defined in SimpleCompiler.h |
| ADDINSTRUCTION | |  | | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| loadInstruction | | int | | Used to create a LOAD instruction to load the first operand of an expression to evaluate into the Accumulator. |
| storeInstruction | | int | | Used to create a STORE instruction to store the result operand of the calculation into the tempStorage address. |
| addInstruction | | int | | Used to create an ADD instruction. |
| subInstruction | | int | | Used to create a SUBTRACT instruction. |
| mulInstruction | | int | | Used to create a MULTIPLY instruction. |
| divInstruction | | int | | Used to create a DIVIDE instruction. |
| expInstruction | | int | | Used to create an EXPONENTIATION instruction. |
| modInstruction | | int | | Used to create a MODULUS instruction. |
| LOAD | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| STORE | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| ADD | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| SUBTRACT | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| DIVIDE | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| MULTIPLY | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| EXPONENTIATION | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| MODULUS | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| MEMORY | | Int array | | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| INSTRUCTIONCOUNTER | | int | | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |

|  |  |  |
| --- | --- | --- |
| **getComparisonOperatorIndex()** | | |
| **Name** | **Type** | **Description** |
| idx | int | Used to get the index of the comparison operator within the token array containing the tokens from a Simple IF statement. |
| idx2 | int | Used to iterate through the compOperators array. |
| compOperators | const char [][] | Used to hold string representations of Simple’s comparison operators. Defined in SimpleCompiler.h. |
| NUMBEROFCOMPOPERATORS | 6 | Used to define the maximum number of comparison operators allowed within the Simple language. Defined in SimpleCompiler.h. |
| FAIL | -1 | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h |

|  |  |  |
| --- | --- | --- |
| **isComparisonOperator()** | | |
| **Name** | **Type** | **Description** |
| idx | int | Used to get the index of the comparison operator within the compOperators array. |
| compOperators | const char [][] | Used to hold string representations of Simple’s comparison operators. Defined in SimpleCompiler.h. |
| NUMBEROFCOMPOPERATORS | 6 | Used to define the maximum number of comparison operators allowed within the Simple language. Defined in SimpleCompiler.h. |
| FAIL | -1 | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h |

|  |  |  |
| --- | --- | --- |
| **doLTJMP()** | | |
| **Name** | **Type** | **Description** |
| LHSAddress | int | Used to hold the address of the result of the expression on the left-hand side of the comparison operator. |
| RHSAddress | int | Used to hold the address of the result of the expression on the right-hand side of the comparison operator. |
| LHSPostfix | char[][] | Used to hold a postfix expression created from the expression on the left-hand side of the comparison operator. |
| RHSPostfix | char[][] | Used to hold a postfix expression created from the expression on the right-hand side of the comparison operator. |
| pfLengthLHS | int | Used to hold the number of tokens added to the Left-hand side postfix array. |
| pfLengthRHS | int | Used to hold the number of tokens added to the Right-hand side postfix array. |
| MAXLINELENGTH |  | Used to limit the length of the postfix expression arrays. Defined in SimpleCompiler.h |
| returnAddress | int | Used to hold the address assigned to the return line number address within the SML Memory space where program execution will branch to. |
| ADDINSTRUCTION |  | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| instruction1 | int | Used to create a LOAD instruction to load the first value of the comparison into the Accumulator. |
| instruction2 | int | Used to create a SUBTRACT instruction to second from the value stored in the Accumulator. |
| instruction3 | int | Used to create a BRANCHNEG instruction to branch the flow of execution to the returnAddress if the value in the Accumulator is negative. |
| returnLineNumber | int | Used to hold the line number of the Simple source code being processed. |
| MEMORY | Int array | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| INSTRUCTIONCOUNTER | int | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| LOAD |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| BRANCH |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| STORE |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| SUBTRACT |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| BRANCHNEG |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| Flags | int [] | Used to hold the SML addresses of instructions which need their operands, eg BRANCH addresses, resolved during the second pass. Defined in SimpleCompiler.h |
| SUCCESS | 1 | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h |
| FAIL | -1 | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h |

|  |  |  |
| --- | --- | --- |
| **doGTJMP()** | | |
| **Name** | **Type** | **Description** |
| LHSAddress | int | Used to hold the address of the result of the expression on the left-hand side of the comparison operator. |
| RHSAddress | int | Used to hold the address of the result of the expression on the right-hand side of the comparison operator. |
| LHSPostfix | char[][] | Used to hold a postfix expression created from the expression on the left-hand side of the comparison operator. |
| RHSPostfix | char[][] | Used to hold a postfix expression created from the expression on the right-hand side of the comparison operator. |
| pfLengthLHS | int | Used to hold the number of tokens added to the Left-hand side postfix array. |
| pfLengthRHS | int | Used to hold the number of tokens added to the Right-hand side postfix array. |
| MAXLINELENGTH |  | Used to limit the length of the postfix expression arrays. Defined in SimpleCompiler.h |
| returnAddress | int | Used to hold the address assigned to the return line number address within the SML Memory space where program execution will branch to. |
| ADDINSTRUCTION |  | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| instruction1 | int | Used to create a LOAD instruction to load the first value of the comparison into the Accumulator. |
| instruction2 | int | Used to create a SUBTRACT instruction to second from the value stored in the Accumulator. |
| instruction3 | int | Used to create a BRANCHPOS instruction to branch the flow of execution to the returnAddress if the value in the Accumulator is greater than zero. |
| returnLineNumber | int | Used to hold the line number of the Simple source code being processed. |
| MEMORY | Int array | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| INSTRUCTIONCOUNTER | int | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| LOAD |  | The value of an OPCode. Defined in SMLOpCodes.h |
| STORE |  | The value of an OPCode. Defined in SMLOpCodes.h |
| BRANCH |  | The value of an OPCode. Defined in SMLOpCodes.h |
| SUBTRACT |  | The value of an OPCode. Defined in SMLOpCodes.h |
| BRANCHPOS |  | The value of an OPCode. Defined in SMLOpCodes.h |
| Flags | int [] | Used to hold the SML addresses of instructions which need their operands, eg BRANCH addresses, resolved during the second pass. Defined in SimpleCompiler.h |
| SUCCESS | 1 | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h |
| FAIL | -1 | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h |

|  |  |  |
| --- | --- | --- |
| **doLTEJMP()** | | |
| **Name** | **Type** | **Description** |
| LHSAddress | int | Used to hold the address of the result of the expression on the left-hand side of the comparison operator. |
| RHSAddress | int | Used to hold the address of the result of the expression on the right-hand side of the comparison operator. |
| LHSPostfix | char[][] | Used to hold a postfix expression created from the expression on the left-hand side of the comparison operator. |
| RHSPostfix | char[][] | Used to hold a postfix expression created from the expression on the right-hand side of the comparison operator. |
| pfLengthLHS | int | Used to hold the number of tokens added to the Left-hand side postfix array. |
| pfLengthRHS | int | Used to hold the number of tokens added to the Right-hand side postfix array. |
| MAXLINELENGTH |  | Used to limit the length of the postfix expression arrays. Defined in SimpleCompiler.h |
| returnAddress | int | Used to hold the address assigned to the return line number address within the SML Memory space where program execution will branch to. |
| ADDINSTRUCTION |  | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| instruction1 | int | Used to create a LOAD instruction to load the first value of the comparison into the Accumulator. |
| instruction2 | int | Used to create a SUBTRACT instruction to second from the value stored in the Accumulator. |
| instruction3 | int | Used to create a BRANCHNEG instruction to branch the flow of execution to the returnAddress if the value in the Accumulator is negative. |
| Instruction4 | int | Used to create a BRANCHZERO instruction to branch the flow of execution to the returnAddress if the value in the Accumulator is equal to zero. |
| returnLineNumber | int | Used to hold the line number of the Simple source code being processed. |
| MEMORY | Int array | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| INSTRUCTIONCOUNTER | int | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| LOAD |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| STORE |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| BRANCH |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| SUBTRACT |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| BRANCHNEG |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| BRANCHZERO |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| Flags | int [] | Used to hold the SML addresses of instructions which need their operands, eg BRANCH addresses, resolved during the second pass. Defined in SimpleCompiler.h |
| SUCCESS | 1 | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h |
| FAIL | -1 | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h |

|  |  |  |
| --- | --- | --- |
| **doGTEJMP()** | | |
| **Name** | **Type** | **Description** |
| LHSAddress | int | Used to hold the address of the result of the expression on the left-hand side of the comparison operator. |
| RHSAddress | int | Used to hold the address of the result of the expression on the right-hand side of the comparison operator. |
| LHSPostfix | char[][] | Used to hold a postfix expression created from the expression on the left-hand side of the comparison operator. |
| RHSPostfix | char[][] | Used to hold a postfix expression created from the expression on the right-hand side of the comparison operator. |
| pfLengthLHS | int | Used to hold the number of tokens added to the Left-hand side postfix array. |
| pfLengthRHS | int | Used to hold the number of tokens added to the Right-hand side postfix array. |
| MAXLINELENGTH |  | Used to limit the length of the postfix expression arrays. Defined in SimpleCompiler.h |
| returnAddress | int | Used to hold the address assigned to the return line number address within the SML Memory space where program execution will branch to. |
| ADDINSTRUCTION |  | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| instruction1 | int | Used to create a LOAD instruction to load the first value of the comparison into the Accumulator. |
| instruction2 | int | Used to create a SUBTRACT instruction to second from the value stored in the Accumulator. |
| instruction3 | int | Used to create a BRANCHPOS instruction to branch the flow of execution to the returnAddress if the value in the Accumulator is greater than zero. |
| Instruction4 | int | Used to create a BRANCHZERO instruction to branch the flow of execution to the returnAddress if the value in the Accumulator is equal to zero. |
| returnLineNumber | int | Used to hold the line number of the Simple source code being processed. |
| MEMORY | Int array | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| INSTRUCTIONCOUNTER | int | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| LOAD |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| STORE |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| BRANCH |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| SUBTRACT |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| BRANCHPOS |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| BRANCHZERO |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| Flags | int [] | Used to hold the SML addresses of instructions which need their operands, eg BRANCH addresses, resolved during the second pass. Defined in SimpleCompiler.h |
| SUCCESS | 1 | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h |
| FAIL | -1 | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h |

|  |  |  |
| --- | --- | --- |
| **doEQJMP()** | | |
| **Name** | **Type** | **Description** |
| LHSAddress | int | Used to hold the address of the result of the expression on the left-hand side of the comparison operator. |
| RHSAddress | int | Used to hold the address of the result of the expression on the right-hand side of the comparison operator. |
| LHSPostfix | char[][] | Used to hold a postfix expression created from the expression on the left-hand side of the comparison operator. |
| RHSPostfix | char[][] | Used to hold a postfix expression created from the expression on the right-hand side of the comparison operator. |
| pfLengthLHS | int | Used to hold the number of tokens added to the Left-hand side postfix array. |
| pfLengthRHS | int | Used to hold the number of tokens added to the Right-hand side postfix array. |
| MAXLINELENGTH |  | Used to limit the length of the postfix expression arrays. Defined in SimpleCompiler.h |
| returnAddress | int | Used to hold the address assigned to the return line number address within the SML Memory space where program execution will branch to. |
| ADDINSTRUCTION |  | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| instruction1 | int | Used to create a LOAD instruction to load the first value of the comparison into the Accumulator. |
| instruction2 | int | Used to create a SUBTRACT instruction to second from the value stored in the Accumulator. |
| Instruction3 | int | Used to create a BRANCHZERO instruction to branch the flow of execution to the returnAddress if the value in the Accumulator is equal to zero. |
| returnLineNumber | int | Used to hold the line number of the Simple source code being processed. |
| MEMORY | Int array | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| INSTRUCTIONCOUNTER | int | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| LOAD |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| STORE |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| BRANCH |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| SUBTRACT |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| BRANCHZERO |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| Flags | int [] | Used to hold the SML addresses of instructions which need their operands, eg BRANCH addresses, resolved during the second pass. Defined in SimpleCompiler.h |
| SUCCESS | 1 | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h |
| FAIL | -1 | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h |

|  |  |  |
| --- | --- | --- |
| **doNEJMP()** | | |
| **Name** | **Type** | **Description** |
| LHSAddress | int | Used to hold the address of the result of the expression on the left-hand side of the comparison operator. |
| RHSAddress | int | Used to hold the address of the result of the expression on the right-hand side of the comparison operator. |
| LHSPostfix | char[][] | Used to hold a postfix expression created from the expression on the left-hand side of the comparison operator. |
| RHSPostfix | char[][] | Used to hold a postfix expression created from the expression on the right-hand side of the comparison operator. |
| pfLengthLHS | int | Used to hold the number of tokens added to the Left-hand side postfix array. |
| pfLengthRHS | int | Used to hold the number of tokens added to the Right-hand side postfix array. |
| MAXLINELENGTH |  | Used to limit the length of the postfix expression arrays. Defined in SimpleCompiler.h |
| returnAddress | int | Used to hold the address assigned to the return line number address within the SML Memory space where program execution will branch to. |
| ADDINSTRUCTION |  | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| instruction1 | int | Used to create a LOAD instruction to load the first value of the comparison into the Accumulator. |
| instruction2 | int | Used to create a SUBTRACT instruction to second from the value stored in the Accumulator. |
| instruction3 | int | Used to create a BRANCHNEG instruction to branch the flow of execution to the returnAddress if the value in the Accumulator is negative. |
| Instruction4 | int | Used to create a BRANCHPOS instruction to branch the flow of execution to the returnAddress if the value in the Accumulator is greater than zero. |
| returnLineNumber | int | Used to hold the line number of the Simple source code being processed. |
| MEMORY | Int array | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h |
| INSTRUCTIONCOUNTER | int | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h |
| LOAD |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| STORE |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| BRANCH |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| SUBTRACT |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| BRANCHNEG |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| BRANCHPOS |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h |
| Flags | int [] | Used to hold the SML addresses of instructions which need their operands, eg BRANCH addresses, resolved during the second pass. Defined in SimpleCompiler.h |
| SUCCESS | 1 | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h |
| FAIL | -1 | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **doSTRCMP()** | | | | | |
| **Name** | | **Type** | | **Description** | |
| Instruction3 | | int | | Used to hold an instruction to be written to the SML Memory space. | |
| Instruction4 | | int | | Used to hold an instruction to be written to the SML Memory space. | |
| returnAddress | | int | | Used to hold the address assigned to the return line number address within the SML Memory space where program execution will branch to. | |
| ADDINSTRUCTION | |  | | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| BRANCHZERO | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h | |
| BRANCHNEG | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h | |
| BRANCHPOS | |  | | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h | |
| INSTRUCTIONCOUNTER | | int | | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h | |
| Flags | int [] | | Used to hold the SML addresses of instructions which need their operands, eg BRANCH addresses, resolved during the second pass. Defined in SimpleCompiler.h | | |
| SUCCESS | 1 | | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | | |
| FAIL | -1 | | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **doSTRCMP()** | | | | |
| **Name** | | **Type** | **Description** | |
| ADDINSTRUCTION |  | | A macro defined in SimpleCompile.h used to improve code readability. Equivalent to;   *if( addInstructionToMemory( instruction ) == FAIL )*  *return FAIL;* |
| loadInstruction | | int | Used to hold an instruction to be written to the SML Memory space. | |
| divideInstruction | | int | Used to hold an instruction to be written to the SML Memory space. | |
| storeInstruction | | int | Used to hold an instruction to be written to the SML Memory space. | |
| multInstruction | | int | Used to hold an instruction to be written to the SML Memory space. | |
| addInstruction | | int | Used to hold an instruction to be written to the SML Memory space. | |
| subInstruction | | int | Used to hold an instruction to be written to the SML Memory space. | |
| branchNegInst | | int | Used to hold an instruction to be written to the SML Memory space. | |
| branchZeroInst | | int | Used to hold an instruction to be written to the SML Memory space. | |
| branchPosInst | | int | Used to hold an instruction to be written to the SML Memory space. | |
| exitInstruction | | int | Used to hold the address of the instruction after ther string compare sequence. | |
| LOAD | |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h | |
| DIVIDE | |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h | |
| STORE | |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h | |
| MULTIPLY | |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h | |
| ADD | |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h | |
| SUBTRACT | |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h | |
| BRANCHNEG | |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h | |
| BRANCHZERO | |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h | |
| BRANCHPOS | |  | The value of an OPCode. It is used to switch to a function creating the necessary instructions for the specified operation. Defined in SMLOpCodes.h | |
| INSTRUCTIONCOUNTER | | int | Used to hold the address with the SML Memory space where the next instruction can be added. Defined in SimpleCompiler.h | |
| MEMORY | | Int array | Used to store all of the instructions, variables and constants necessary to running an SML program. Defined in SimpleCompiler.h | |
| VARCONSTADDRESS | | int | Used to hold the next memory address, at the end of the SML Memory space, where a value can be added. Defined in SimpleCompiler.h | |
| addressA | | int | Used to hold the address in MEMORY of the first string. | |
| startOfA | | int | Used to hold the address in MEMORY where a copy of the start address of the first string is stored. | |
| positionInA | | int | Used when iterating through the first string in MEMORY. | |
| addressB | | int | Used to hold the address in MEMORY of the second string. | |
| startOfB | | int | Used to hold the address in MEMORY where a copy of the start address of the second string is stored. | |
| positionInB | | int | Used when iterating through the second string in MEMORY. | |
| one | | int | Used to holds the address in MEMORY where a constant value of 1 is stored. | |
| thousand | | int | Used to holds the address in MEMORY where a constant value of 1000 is stored. | |
| loadA | | int | Used to create an instruction to dynamically LOAD string A into MEMORY. | |
| subB | | int | Used to create an instruction to dynamically SUBTRACT string B from the ACCUMULATOR into MEMORY. | |
| strlenAThousand | | int | Used to extract the first character from the first string. | |
| maxA | | int | Used to hold the end address of the space allocated to the first string in MEMORY. | |
| tempA | | int | Used to temporarily hold the address of the first character in the first string to be compared against the first character in the second string. | |
| strlenBThousand | | int | Used to extract the first character from the second string. | |
| SUCCESS | | 1 | Used as a return value for a successful operation to make the code more readable.  Defined in SimpleCompiler.h | |
| FAIL | | -1 | Used as a return value for a failed operation to make the code more readable.  Defined in SimpleCompiler.h | |