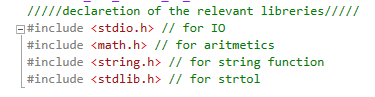
Assembler

The assembler written in C language and its convert the SIMP assembly language into a machine language in which each line contains 5 characters in the base 16.

The input (“asm\_program”) of the assembler is a file that contains a SIMP assembly language and the output (“memin”) is a text file containing the corresponding machine language.

The following are the steps of the assembler:

1. Declaration of the libraries relevant to the Code.



1. Definition of given global variables.

A screenshot of a computer

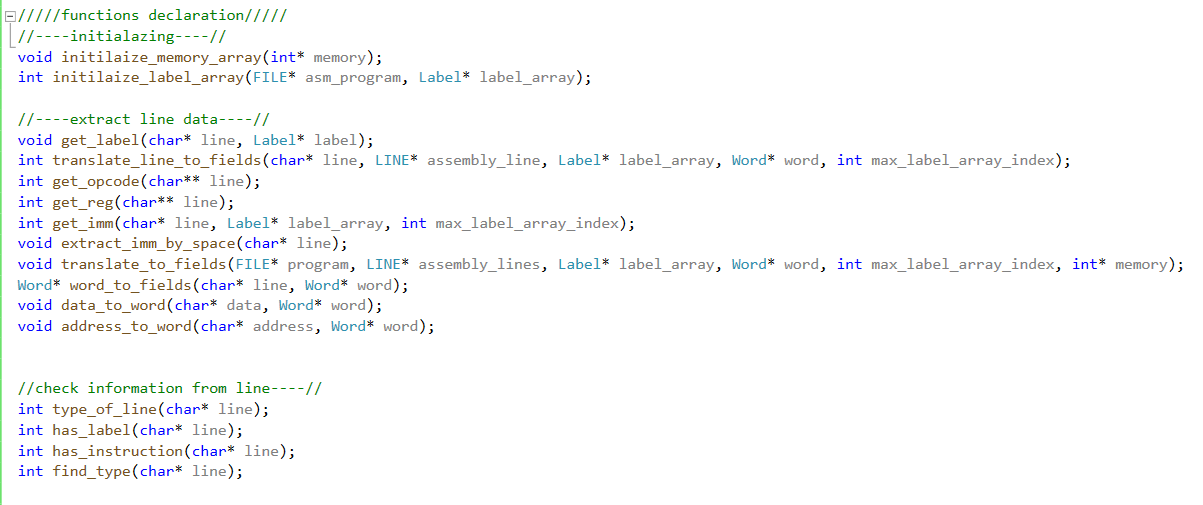
Description automatically generated with medium confidence

1. Definition of 3 data structures, Label, LINE and Word.

Text

Description automatically generated

1. Declaration of functions we will use.

Graphical user interface, text, application

Description automatically generated

1. The main function.

Text

Description automatically generated

* 1. Opening two files and checking that the operation was successful. “asm\_program” for reading and “memin” for writing.
  2. Initialization of major variables:
     1. An array of characters named "label\_array" (Label types) size 4096.
        1. Initialization function: initilaize\_label\_array(FILE\* asm\_program, Label\* label\_array).
        2. Go through the input file “asm\_program” from the beginning till the end, using a while loop. Checks the row type, if the row includes a label, inserting the label and its address into the array. The label address is determined by a variable that is updated each read by row type. Rtype increases by 1, Itype increases by 2, and a label, the ".word" command or empty row do not increase the variable.
        3. Note: we use calloc to allocate memory to this array.
     2. Variable named “word”, a Word Struct.
     3. A 4096-sized array of numbers named “memory” that will contain the rows to be printed to the output file in the appropriate location in the array.
        1. Initialization function: initilaize\_memory\_array(int\* memory).
        2. Insert the string "00000" into every value in the array.
        3. Note: we use calloc to allocate memory to this array.
     4. An array of rows named "assembly\_lines" (LINE types) of size 4096.
        1. Initialization function: translate\_to\_fields(FILE\* program, LINE\* assembly\_lines, Label\* label\_array, Word\* word, int max\_label\_array\_index, int\* memory).
        2. Go through the input file "asm\_program" from beginning till the end using a while loop. Erasing comments from the raw and extract the value of each respective field in the LINE type (opcode, rd, rs, rt, imm). converts the field values to a 5-character base 16 number and save it in the appropriate place in the “memory” array. If the row contains an Itype command, we update the “memory” array with the value of the imm into the appropriate place in the array as well. If the line contains a ".word" command, we insert the data value into the address place in the memory array.
        3. Note: we use calloc to allocate memory to this array.
  3. Writing the text in the output file "memin":
     1. Writing function: write\_output\_to\_file(FILE\* output, int\* memory).
     2. Go through all the “memory” array’s values using a For loop and print them in the output file "memin" line by line corresponding to their location in the array.
  4. Closing the files:
     1. Closing function: close\_files(FILE\* asm\_program, FILE\* memin).
     2. Closes both files, input “asm\_program” and input “memin”.
  5. Free memory allocation:
     1. Free memory function: free\_allocate\_memorry(Label\* label\_array, LINE\* assembly\_lines, int\* memory)
     2. Free the memory allocation of the label array “label\_array”, the array of the struct LINE “assembly\_lines” and the array of int “memory”.