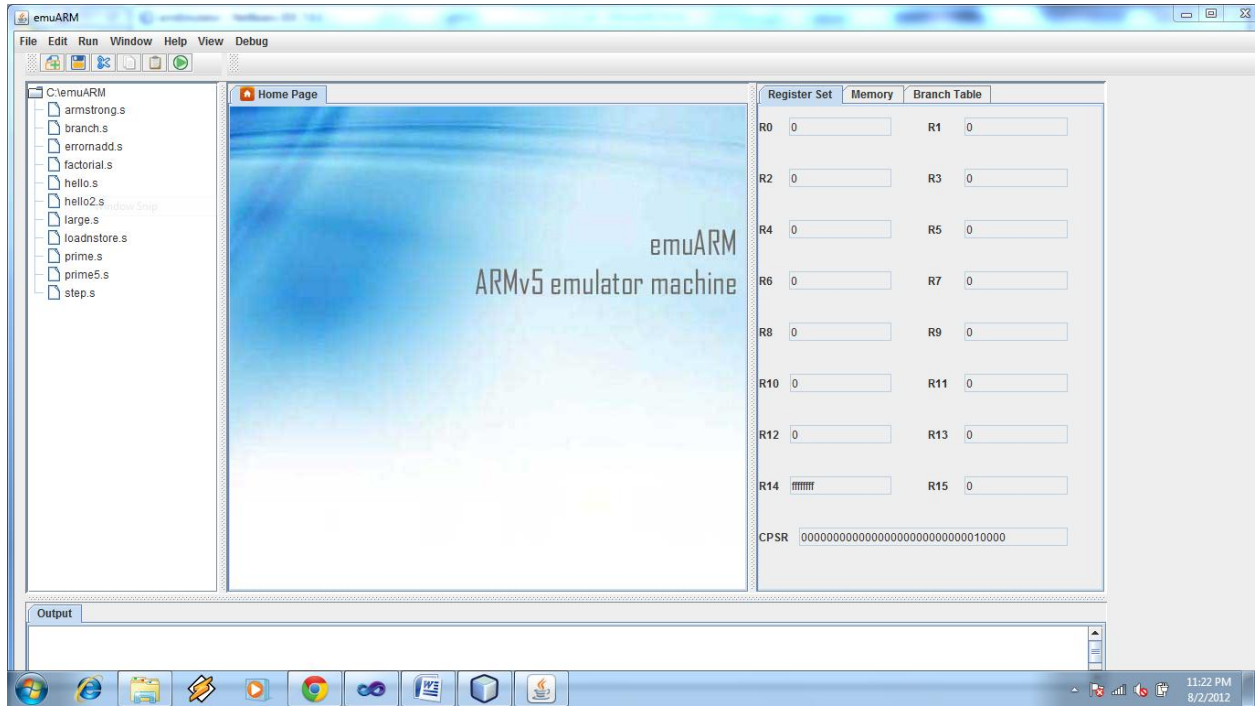


Run the jar file to launch the Application.

## Main Window

The following is the main screen:



The above is the home screen of the application. There is a tabbed editor pane, a menu bar, a title bar, a tool bar, tree view of the current working directory, status dialog box and other than that there are register status display pane, memory status display pane and branch table display pane.

## Register Set status

This is a display pane that shows the updates done on the 16 registers present in the user mode of the processor. Along with that it shows the contents of CPSR register which is Current Program Status Register.

[illegible]

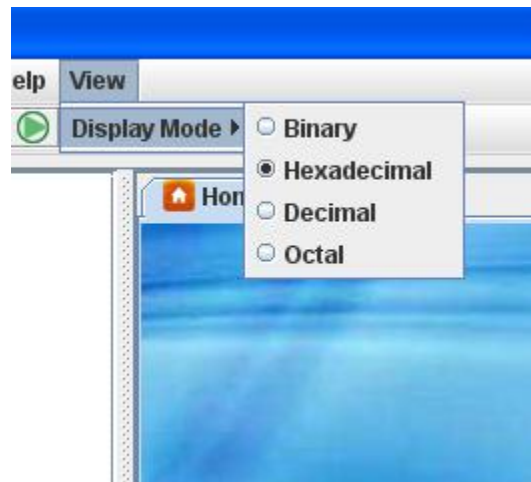
There is a provision of choice of language for display of registers. It is human mind set to follow decimal number system. But as we know that machine language is made up of 0s and 1s. Computers follow binary number system. In this emulator machine, by default the data that is displayed will be in hexadecimal number system. But if we wish to see the data in

decimal or binary then, we can switch to that mode. Changing to other mode will convert all register values to the selected mode.

Other options for display are:

- a. Decimal
- b. Octal
- c. Hexadecimal
- d. Binary

To switch from one mode to another, we can change from the Display mode of the view menu. Following screenshot displays that option:



The selection from these radio buttons is mutually exclusive in nature i.e, we cannot have a display mode to be both binary and decimal at the same time. Following screenshots shows the data in different number systems:

- In Hexadecimal

Register Set		Memory	Branch Table	
R0	d		R1	9
R2	0		R3	4
R4	0		R5	0
R6	0		R7	0

- In Binary

Register Set		Memory	Branch Table	
R0	1101		R1	1001
R2	0		R3	100
R4	0		R5	0

- In Octal

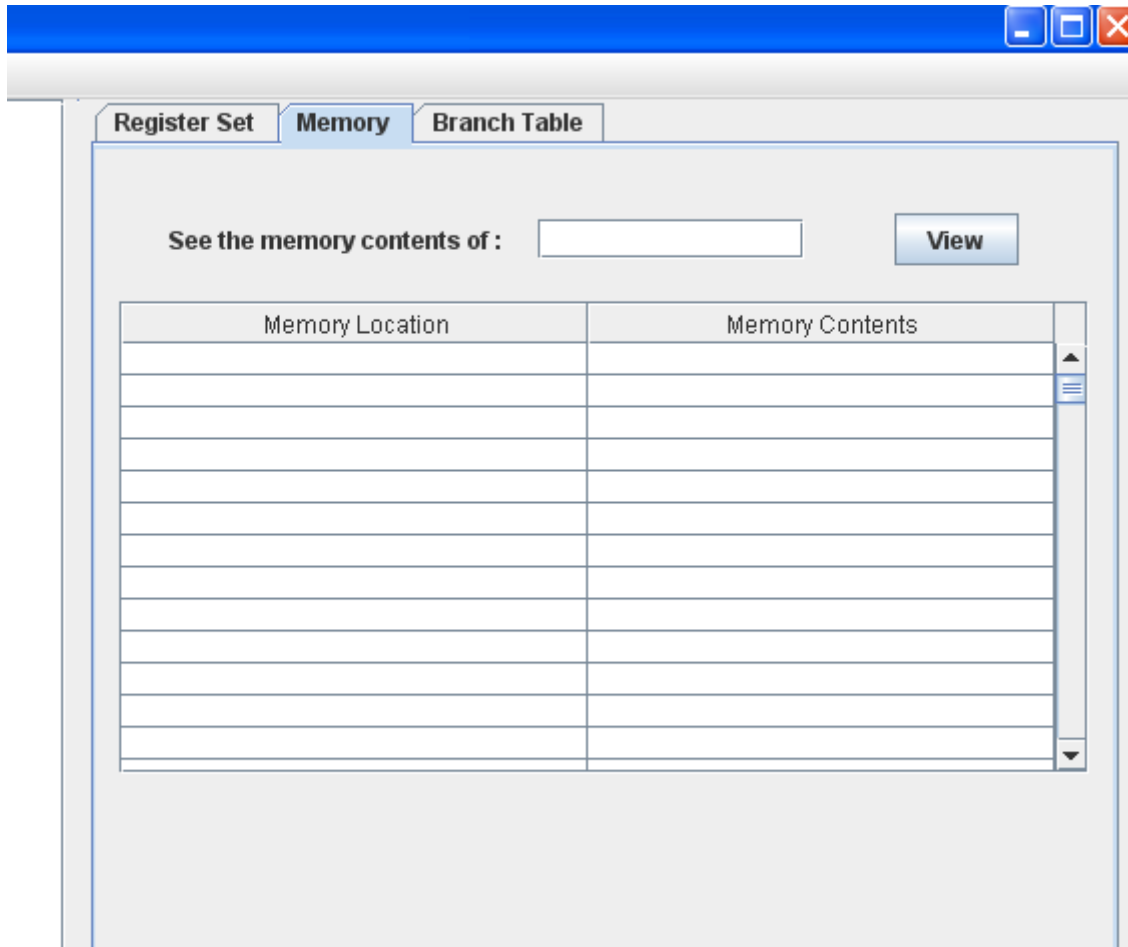
Register Set		Memory	Branch Table	
R0	15		R1	11
R2	0		R3	4
R4	0		R5	0

- In Decimal

Register Set		Memory	Branch Table	
R0	13		R1	9
R2	0		R3	4
R4	0		R5	0

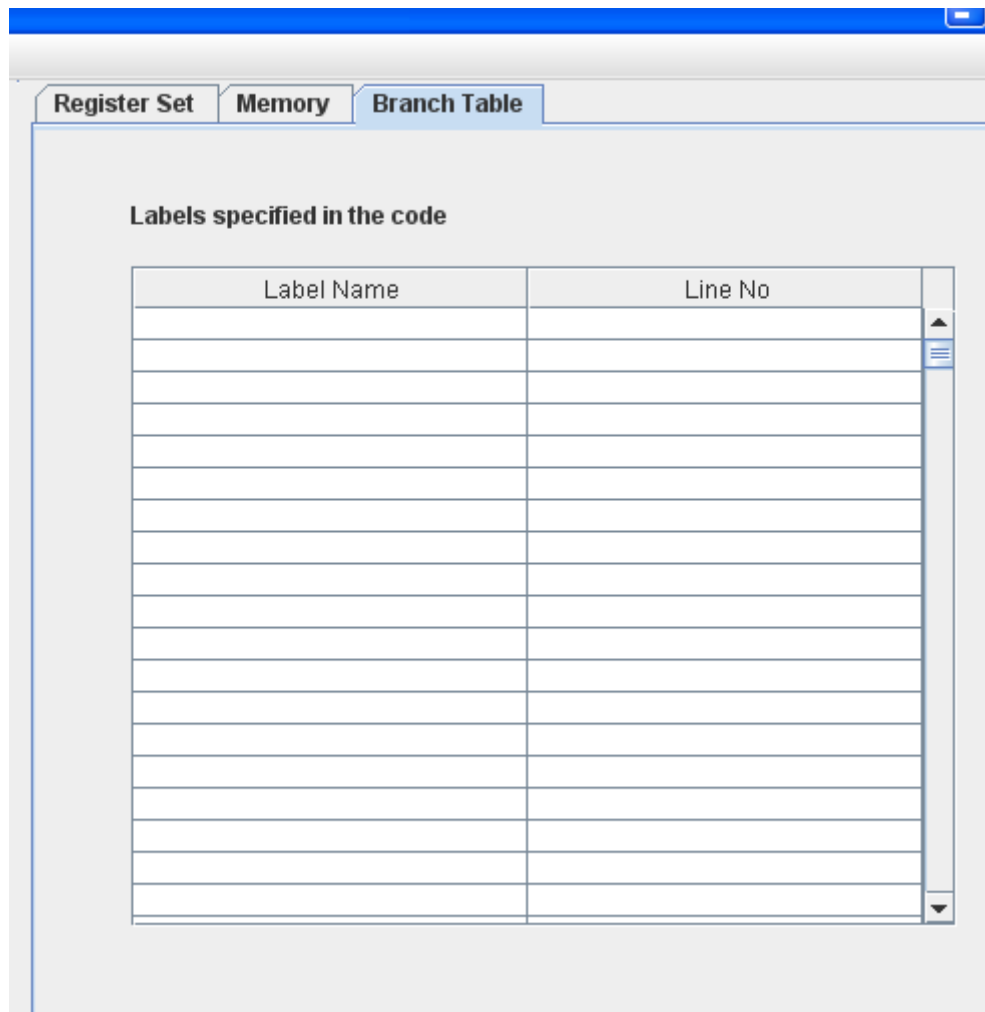
## **Memory status**

This display pane shows the tabular form of the memory module. It shows the contents of the active memory locations. The contents are shown only on the basis of the active program.



## **Branch table**

This pane displays the branch table which is formed during the first pass. It shows all the labels which are present on their respective line numbers. The branch table helps in branching or relocating the position pointer of the execution of the program to other random places depending upon the B, BL instructions.

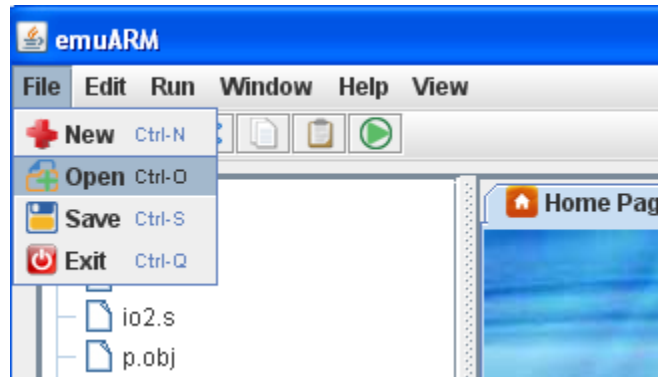



## **Menu Bar**


The menu bar of the main window is having seven menus, namely, File, Edit, Window, Help, View, Debug and Run. The options in the menus of menu bar are as follows:




## File Menu



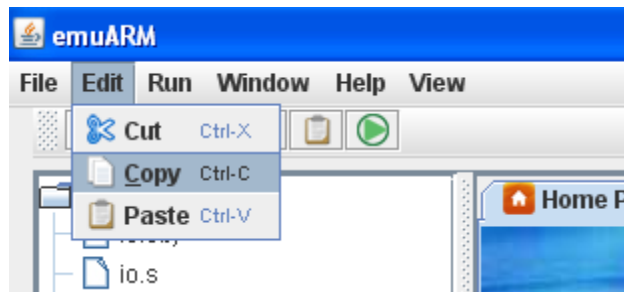
 **Exit** Ctrl-Q : **Exit** – It closes the application


 **New** Ctrl-N : **New** – It provides a new empty pane to enter the code


 **Open** Ctrl-O : **Open** – Open a pre-saved assembly file

 **Save** Ctrl-S : **Save** – Save an assembly file (.S)


## Edit Menu



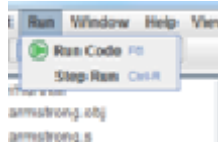
 **Cut** Ctrl-X : **Cut** – Moves a section of code fragment to the clipboard


 **Copy** Ctrl-C : **Copy** – Copies a section of code fragment to the clipboard




 **Paste** Ctrl-V : **Paste** – Copies data from clipboard to the editorpane

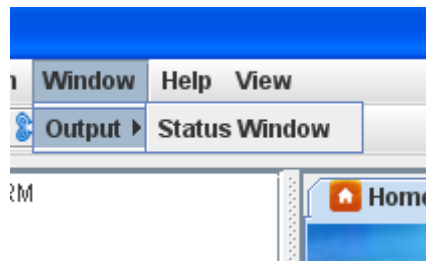
## Run Menu



 **Run Code** F6 : **Run Code** – Execute the code opened in the editorpane

 **Step Run** Ctrl-R : **Step Run** – Execute the code step by step (Step Into)

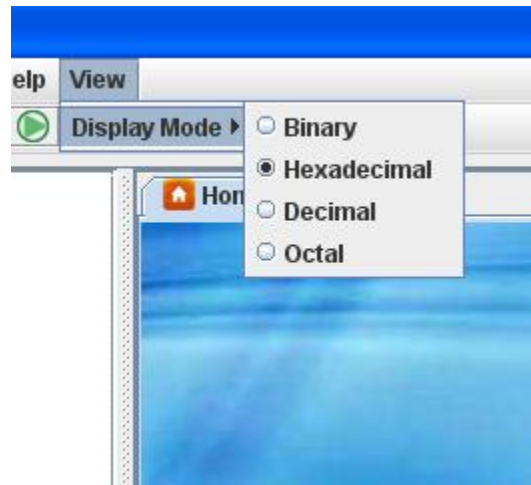
## Window Menu



 **Status Window** : **Status Window** – Display the status window in a separate window

## **View Menu**

Following screenshot displays the View menu that is introduced in the Menu bar:



View menu contains the display mode option from where we can switch our display from the decimal mode to the binary or any other required mode for displaying the contents of registers.

- In Hexadecimal

Register Set		Memory	Branch Table	
R0	d		R1	9
R2	0		R3	4
R4	0		R5	0
R6	0		R7	0

- In Binary

Register Set		Memory	Branch Table	
R0	1101		R1	1001
R2	0		R3	100
R4	0		R5	0

- In Octal

Register Set		Memory	Branch Table	
R0	15		R1	11
R2	0		R3	4
R4	0		R5	0

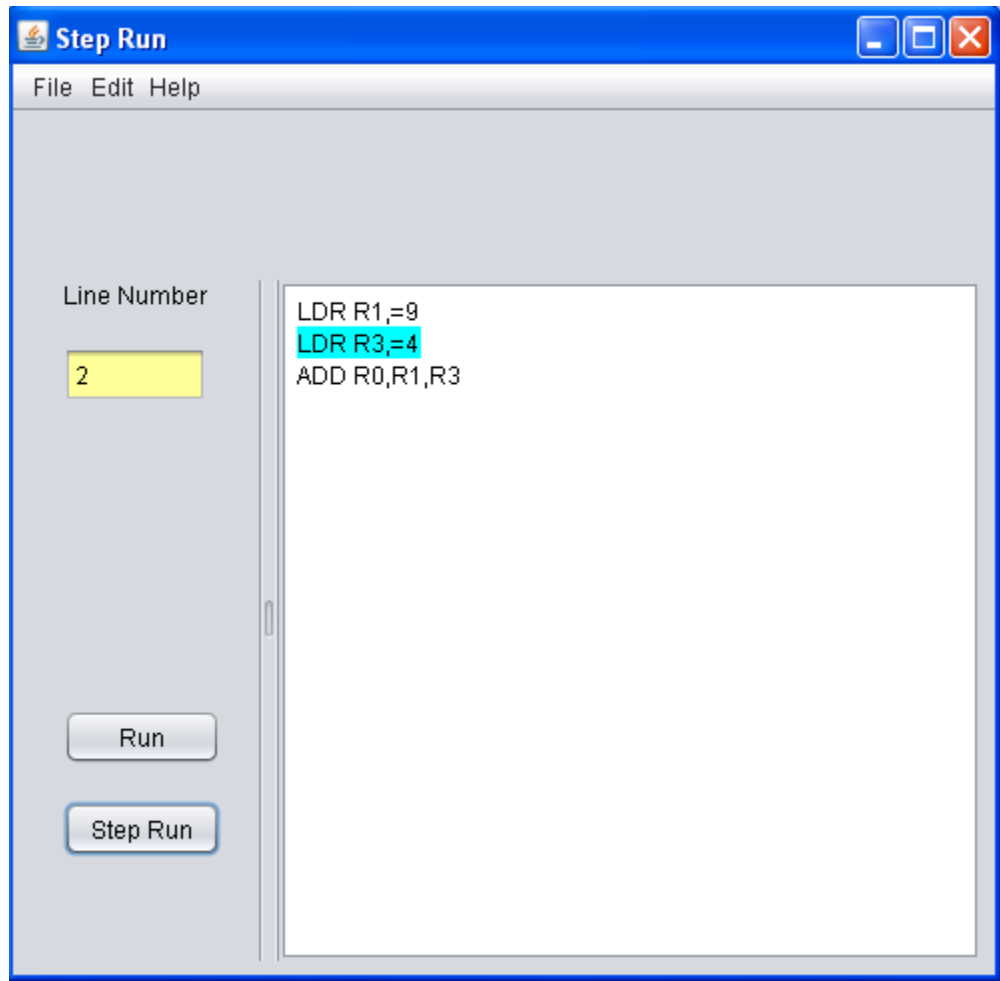
- In Decimal

Register Set		Memory	Branch Table	
R0	13		R1	9
R2	0		R3	4
R4	0		R5	0

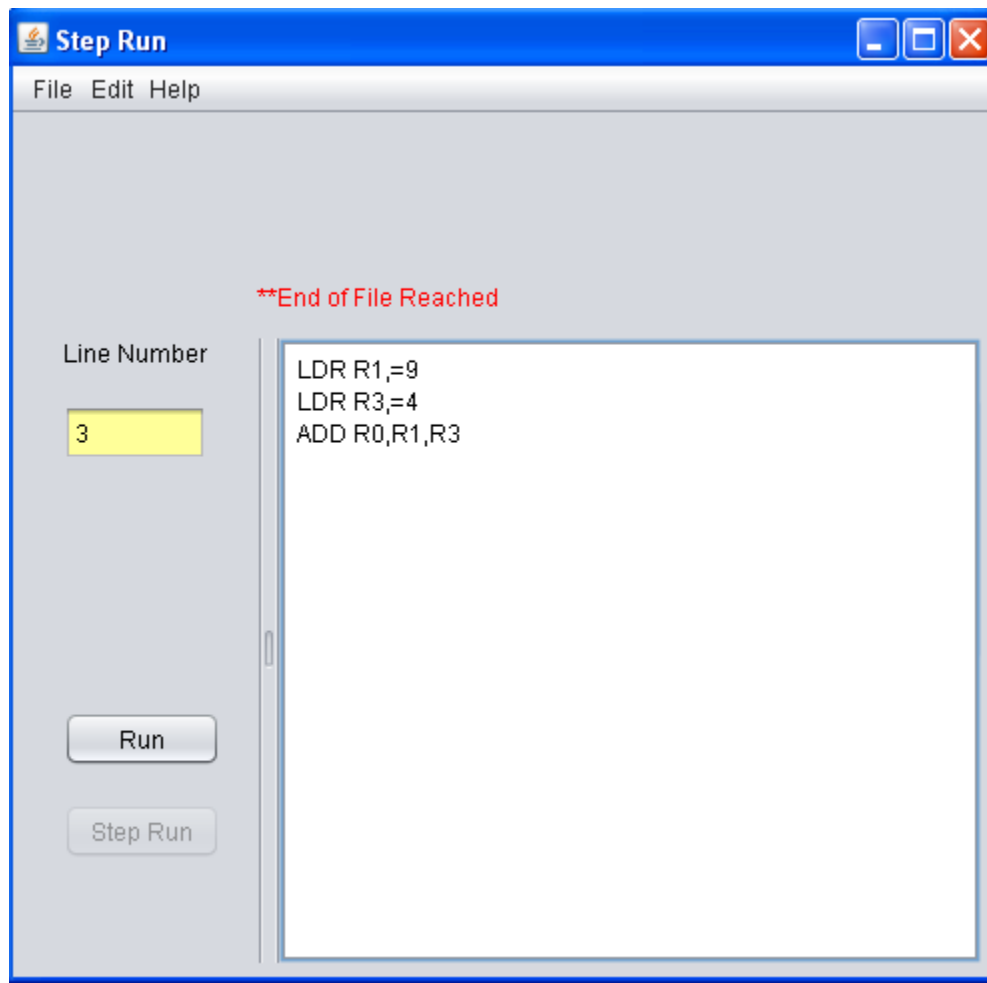
### **Step Into**

If there is a mistake (or "bug") in a program, it is often quite hard for the programmer to track down the exact cause. The process of finding "bugs" in a program is referred to as "debugging" and can be one of the hardest tasks in programming.

Following screenshot displays the step run window:

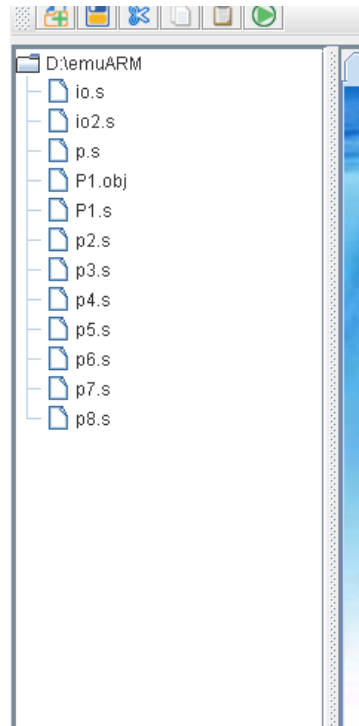


In this window, current line which is executed is highlighted as well its number is displayed in the text box. We can execute the program line by line from this window. Along with step run, at any moment, we can continue with the normal running of the program using run button. After all the lines have completed with the execution, the button is disabled and a message is displayed on the window that the end of file is reached. Following screenshot illustrates that:



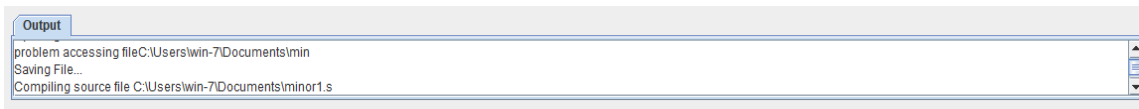
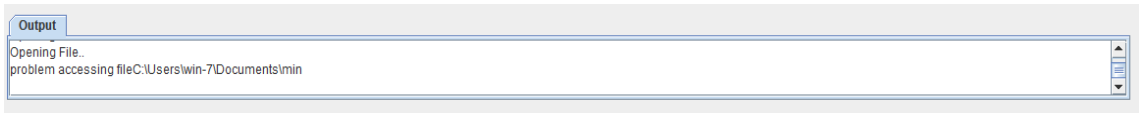
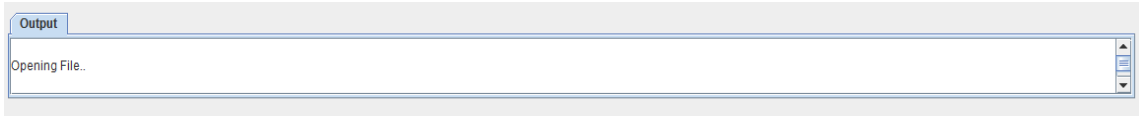
## **Tree view**

It tells about the files present in the current working directory. Files could be opened directly from that directory structure. This is a nice way to interactively open files and display the current files that are present in the working folder

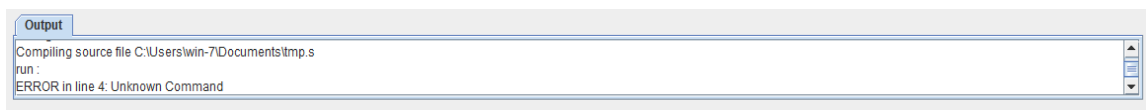


## **Status window**

The status window of the user interface tells about the task that the emulator is performing like when the emulator is opening a file or saving any file. It tells when any error occurs while accessing the file for opening. All the error listing developed by the error handling module are listed on this status window. Hence, this status window has multiple functions. In case any error arises while running the file, it tells us that the build has failed and gives the list of errors along with the line numbers. Below is its screenshot.

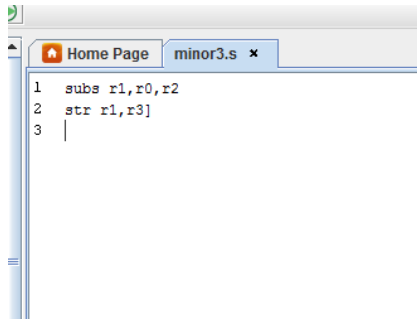


We come across syntactical errors while parsing. In this project, when we catch errors due to improper syntax, we raise exceptions and show the descriptive error details on the status window. In this way, the user gets an idea about the line number of the error along with a hint about the nature of the error. Following screenshots illustrate the way status bar show the nature of the syntactical errors:

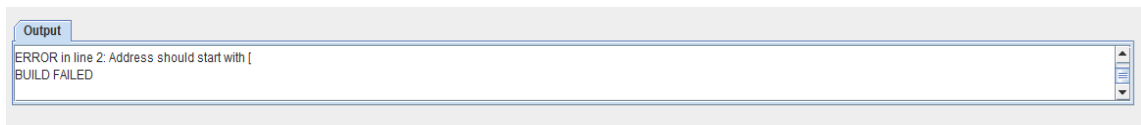


Here, line numbers come up along with the nature of the error. Like for example every addressing mode of an operand should start with '[' symbol failing which incurs an error. Following screenshot illustrate that error:



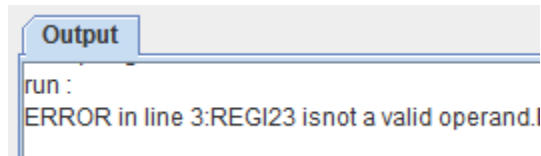


```
1  subs r1,r0,r2
2  str r1,r3]
3  |
```



Output  
ERROR in line 2: Address should start with [  
BUILD FAILED

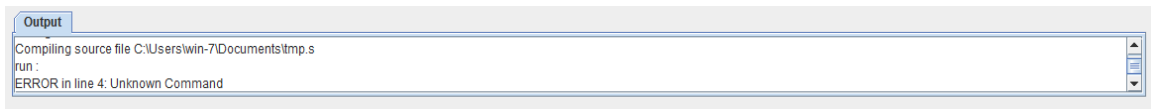
Similarly, other cases are taken care of. Several kinds of exceptions helped in finding the errors. `IndexOutOfBoundsException` helped in catching an error when a program used any register out of its given 0-15 range. Following screenshot illustrates that:



Output  
run :  
ERROR in line 3:REGI23 isnot a valid operand.I

Error handling started in initial stages only. In the backend, when the program is scanned line by line and the scanned line is tokenized. Every token of the line is checked if it is valid or not. The very first check is done at the time of emitting the command to its proper interface

by the hashtable. Now, if the command is not a valid one than the control does not move further for the execution of the scanned line. Error is caught and is listed on the status window. Following screenshot illustrates the same:



At the time of debugging, we tend to introduce print statements to get the exact location where the error has occurred. To provide this facility, input output module is built which has printf and scanf facility.

Note that the status window in the home screen is quite small in width and a lot of scrolling is required at times. In such situations, it is better to view the status window in a different dialog box. Following screenshot shows that dialog box:

