Keyboard functions in Jack

A PROJECT REPORT
Submitted by – Batch A TEAM_7

SUBMITTED BY

01 M.PRASANNA TEJA (CB.EN.U4AIE21035)

02 G. HIMAMSH (CB.EN.U4AIE21014)

03 G.SRI VATSANKA (CB.EN.U4AIE21010)

04 VIKHYAT BANSAL (CB.EN.U4AIE21076)

ELEMENTS OF COMPUTING-II

AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY UNDER SECTION 3 OF THE UGC ACT, 1956

Centre for Computational Engineering and Networking

AMRITA SCHOOL OF ENGINEERING

AMRITA VISHWA VIDYAPEETHAM
COIMBATORE - 641 112

SUPERVISED BY – Prof.Sreelakshmi K (Asst.Professor)

TABLE OF CONTENT

1)AIM

2)ACKNOWLEDGEMENT

3)DECLARATION

4)ABSTRACT

5)INTRODUCTION

6)JACK LANGUAGE

6.1- Jack OS 6

6.2-Standard Library. 7

6.3-Keyboard class 8

6.3.1-Conversions 9

7)ASCII CODE

7.1- Theory 9

7.2-ASCII Table 10

8)Code

8.1- Keyboard Code 11

8.2-Main Jack Code 13

9)Implementation

10) Extras

11)Conclusion

AIM

Implementing the keyboard library in Jack:

1. Returns the ASCII code (as char) of the currently pressed key, or 0 if no key is currently pressed.

2. Selects the following key from the keyboard: awaits the moment a key is pressed before the key is released, after which it is echoed to the screen and its value is returned.

3. Prints the message on the screen, reads the next line (until a newline character) from the keyboard, and returns its value.

4.Prints the message on the screen, reads the next line (until a newline character) from the keyboard, and returns its integer value (until the first nonnumeric character

ACKNOWLEDGEMENT

We are deeply thankful to Center for Excellence in **Computational Engineering and Networking (CEN) at Amrita Vishwa Vidyapeetham, Coimbatore** for providing us such a wonderful environment to peruse our research. We would like to express our sincere gratitude to **Prof.SreeLakshmi. K (Asst. Prof) Department of CEN, Amrita Vishwa Vidyapeetham Coimbatore**. We have completed our research under her guidance we found the research area, topic and problem with her suggestions she guided us with our study and supplied us as many research papers and academic resources in this area.

We would also like to extend our gratitude to
Jyothish Lal.G (Asst.prof) department of CEN, Amrita Vishwa Vidyapeetham Coimbatore who provided us the basic knowledge of this field of this research. we would also like to acknowledge our team members for supporting each other and be grateful to university for providing this opportunity.

Finally, we would like to thank Noam Nisan, Shimon and their team for providing us wonderful tool and course named **Nand2tetris** which helped us a lot in research work.

DECLARATION

We, **The Team-7 of Batch-A** declare that the work embodied in this project work hereby, titled "**Keyboard functions in Jack**", forms our team contribution to the research work carried out under the guidance of **Prof.SreeLakshmi. K (Asst. Prof) Department of CEN, Amrita Vishwa Vidyapeetham Coimbatore.**

It is a result of our team research work and has not been previously submitted to any other University for any other Degree/ Diploma to this or any other University.

Wherever reference has been made to previous works of others, it has been clearly indicated as such and included. We, here by further declare that all information of this document has been obtained and presented in accordance with academic rules and ethical conduct. So, we hope you will like the project which we had done.

From groupmates of Team-7,

M. Prasanna Teja (CB.EN.U4AIE21035)

G. Himamsh (CB.EN.U4AIE21014)

G. Sri Vatsanka (CB.EN.U4AIE21010)

Vikhyat Bansal (CB.EN.U4AIE21076)

Place: Ettimadai
Date: XX-7-2022

ABSTRACT

- We must develop a Jack code and use the Jack Compiler to compile the Jack file to VM file. Later, the VM file will be translated to an assembly language code file, which is abbreviated as ASM.

- After developing the ASM file, that ASM file will be converted to a HACK file with help of ASSEMBLER.

- The appropriate files, such as VM, ASM, and hack files, must be examined in the appropriate platforms, such as VM emulator, CPU emulator, hardware simulator, and assembler. The nand2tetris webpages are the source of this program.

Required Steps:

- Step-1: We will convert .Jack files to .VM files.
- Step-2: We will convert .Jack files to .VM files.
- Step-3: We will convert .Jack files to .VM files.

Introduction

1. Keyboard functions are provided, and we are requested to create a Jack programme that completes the task, compiles, and runs successfully on a VM emulator.

2. High-level programmers can be created using Jack, a straightforward object-based language.

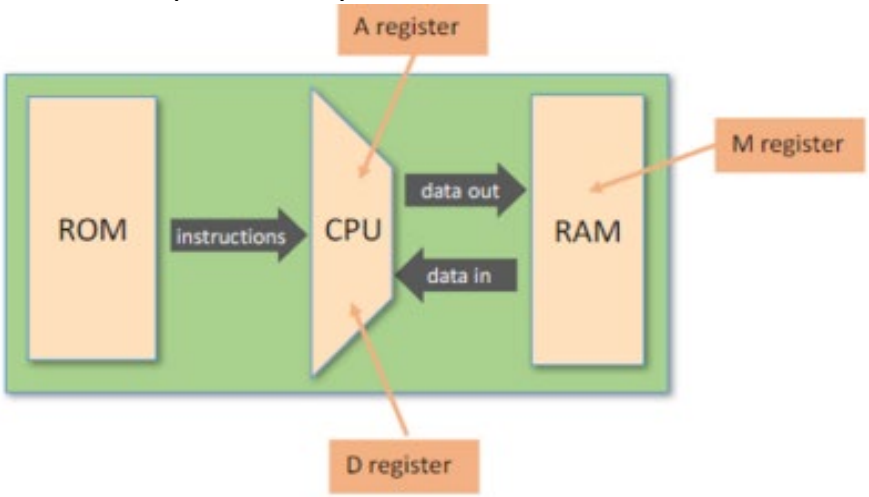
3. We have to write Jack code, and with help of Jack Compiler, we have to compile the jack file to VM file, later VM file is to be converted to assembly language code file, simply called ASM file.

4. Finally, that assembly code file is to be converted to 16- bit machine code i.e., hack file with the help of an assembler.

5. We have to make computer chip. Then to check hack files in hardware simulator we have to load computer chip first.

6. The final hack is to be loaded in the computer chip in hardware simulator. We have to check the respective files, like VM, ASM file, hack file in the respective platforms like VM emulator, CPU emulator, hardware simulator, assembler.

7. This software is provided by the nand2tetris websites.



6.JACK LANGUAGE

6.1-JackOS:

1. Writing Jack programs requires working with the Jack OS, just like writing Java programs requires working with the Java class library.

2. The Jack OS can be viewed as a collection of software services that extend the basic language's capabilities and close gaps between it and the underlying hardware. Here is the Jack OS API.

3. The Nand2tetris Software Suite includes two Jack OS implementations.

4. One OS implementation was written in Jack, and was then translated using a Jack compiler into the set of 8 class files Math.vm, Screen.vm, Output.vm, Keyboard.vm, Memory.vm, String.vm, Array.vm, and Sys.vm.

5. We have to further study on library of Keyboard which will be discussed further.

6. These files are stored in the nand2tetris/tools/os folder in your computer.

7. The second OS implementation was written in Java, and is an integral part of the supplied VM emulator, also available in nand2tetris/tools.

8. Below we explain how to use each OS version.

6.2-Jack standard Library:

- The jack programming language includes a standard library which can be seen as an interface to an underlying operating system.
- The library is a set of jack classes that must be included in every jack implementation.
- The following classes are included in a standard Library:

1. **Math**: Provides basic Mathematical operations.
2. **String**: Implements the string type & basic string-related operation
3. **Array**: Defines array type, array construction & disposal of array
4. **Output**: Handles text-based output.
5. **Screen**: Handles Graphic screen output.
6. **Keyboard**: Handles user input from the keyboard
7. **Memory**: Handles memory operations.
8. **Sys**: Provides some execution-related services

6.3-Keyboard classes:

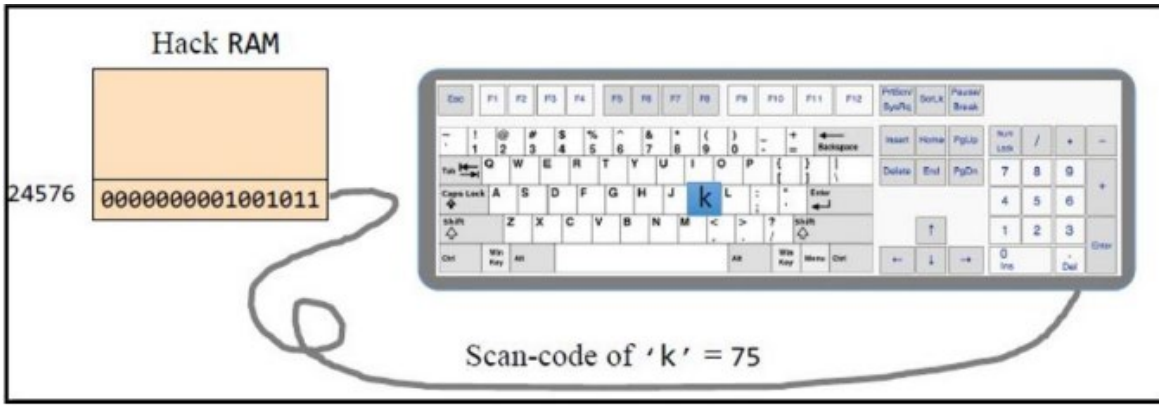
This class allows reading the input from the keyboard.

1. **Function char keyPressed():** Returns the character of the currently pressed key on the keyboard ;if no key is currently pressed, returns 0.

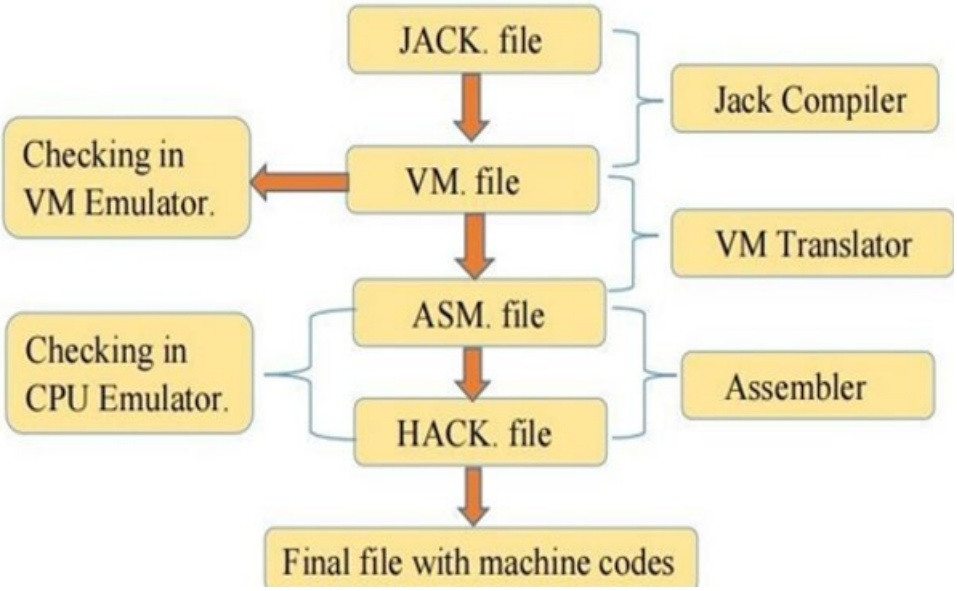
2. **Function char readChar():** Waits until a key is pressed on the keyboard and released and echoes the key to the screen and returns the character of the pressed key.

3. **Function string readLine(string message):** Prints the message on the screen, reads the next line from the keyboard ,echoes the line to the screen, and returns it's value. This method handles user backspaces.

4. **Function int readInt(string message):**Prints the message on the screen, reads the next line from the keyboard ,echoes the line to the screen, and returns the integer until the first non-numeric character in the line. This method handles user backspaces.



6.3.1 - Conversion of files:



ASCII CODE

- ASCII stands for "**American standard code for information interchange**".
- ASCII is 7-bit character set containing 128 character.
- It contains numbers from 0-9 ,the upper and lower cases English letter from A to Z and some special character(!,@,#,%,*...etc).
- The characters are used in modern computers ,Html and on internet are based on ASCII code.

ASCII Table

```
/**
 * Returns the ASCII code (as char) of the currently pressed key,
 * or 0 if no key is currently pressed.
 * Recognizes all ASCII characters, as well as the following extension
 * of action keys:
 * New line = 128 = String.newLine()
 * Backspace = 129 = String.backSpace()
 * Left Arrow = 130
 * Up Arrow = 131
 * Right Arrow = 132
 * Down Arrow = 133
 * Home = 134
 * End = 135
 * Page Up = 136
 * Page Down = 137
 * Insert = 138
 * Delete = 139
 * ESC = 140
 * F1 - F12 = 141 - 152
 */
```

A	65	41	a	97	61
B	66	42	b	98	62
C	67	43	c	99	63
D	68	44	d	100	64
E	69	45	e	101	65
F	70	46	f	102	66
G	71	47	g	103	67
H	72	48	h	104	68
I	73	49	i	105	69
J	74	4A	j	106	6A
K	75	4B	k	107	6B
L	76	4C	l	108	6C
M	77	4D	m	109	6D
N	78	4E	n	110	6E
O	79	4F	o	111	6F
P	80	50	p	112	70
Q	81	51	q	113	71
R	82	52	r	114	72
S	83	53	s	115	73
T	84	54	t	116	74
U	85	55	u	117	75
V	86	56	v	118	76
W	87	57	w	119	77
X	88	58	x	120	78
Y	89	59	y	121	79
Z	90	5A	z	122	7A
[	91	5B	{	123	7B
\	92	5C	|	124	7C
]	93	5D	}	125	7D
^	94	5E	~	126	7E
_	95	5F	[DEL]	127	7F

Keyboard jack code

TERM_PROJECT_SEM_2 > Keyboard.jack

```
class Keyboard {
    static Array keyboard;
    field int input;


    /** Initializes the keyboard. */
    constructor Keyboard new(){
        return this;
    }

    function void init() {
        let keyboard = 24576;
        return;
    }


    function char KeyPressed() {
        return keyboard[0];
        }

    function char readChar() {
        var char k;
        while( Keyboard.KeyPressed() = 0 ) {}
            let k = Keyboard.KeyPressed();

        while( ~(Keyboard.KeyPressed() = 0) ) {}

        return k;
    }
```

```
30
31      function String readLine(String text) {
32
33          var String val;
34          var char c;
35
36          let val =String.new(69);
37          let c = 0;
38
39          do Output.printString(text);
40
41          while (~(c=128)) {
42              let c = Keyboard.readChar();
43              if (c=129){
44                  do val.eraseLastChar();
45                  do Output.printChar(129);
46              }
47              else{
48                  do Output.printChar(c);
49                  do val.appendChar(c);
50              }
51          }
52          do val.eraseLastChar();
53          return val;
54      }
```

```
55
56      function int readInt(String message) {
57
58          var String val;
59          var Int ival;
60          var char c;
61
62          let val =String.new(69);
63          let c = 0;
64
65          do Output.printString(message);
66
67          while (~(c=128)) {
68              let c = Keyboard.readChar();
69              if (c=129){
70                  do val.eraseLastChar();
71                  do Output.printChar(c);
72
73              }
74              else{
75                  do val.appendChar(c);
76                  do Output.printChar(c);
77              }
78          }
79          do val.eraseLastChar();
80          let ival=val.intValue();
81          return ival;
82      }
83
84  }
85
```

Compiled keyboard code: -

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\HP\Desktop\Amrita\AMRITA APPS\nand2tetris\nand2tetris\tools> .\JackCompiler.bat .\TERM_PROJECT_SEM_2\Keyboard.jack
Compiling "C:\Users\HP\Desktop\Amrita\AMRITA APPS\nand2tetris\nand2tetris\tools\TERM_PROJECT_SEM_2\Keyboard.jack"
PS C:\Users\HP\Desktop\Amrita\AMRITA APPS\nand2tetris\nand2tetris\tools>
```

Main Jack code: -

TERM_PROJECT_SEM_2 > Main.jack

```
class Main {
    function void main(){
        var int Keys;
        var int input;
        var Keyboard keyboard;
        var boolean run;
        var String a;
        var String s;
        var int integer;

        // let keyboard = Keyboard.new();

    let run = false;
    do Output.printString("PROGRAM FOR READING CHARACTERS FROM THE KEYBOARD.");
    do Output.println();
    do Output.printString("Please press 'ESC' to move to the next step.");
    do Output.println();
    do Output.println();
    do Output.printString("Keep pressing any key to start.");
    do Output.println();
    do Output.println();
    let Keys = Keyboard.KeyPressed();
        do Sys.wait(1000);
        do Output.printString("The ASCII of currently pressed key is: ");
        do Output.printInt(Keys);
        do Output.println();
```

```

    while(~run) {

        do Sys.wait(500);
        do Output.println();
        let input = Keyboard.readChar();
        do Output.println();
        do Sys.wait(100);
        do Output.printString("New character pressed is: ");
        do Output.printChar(input);
        do Output.println();
        do Sys.wait(100);
        do Output.printString("The ASCII of this character is: ");
        do Output.printInt(input);
        do Output.println();


        if (input = 140) {
            do Sys.wait(100);
            do Output.println();
            do Output.printString("Moving to read string from user.");
            do Sys.wait(1000);
            do Screen.clearScreen();
            do Output.moveCursor(0,0);
            let run = true;
```

TERM_PROJECT_SEM_2 > Main.jack

```
52          }
53
54      }
55
56          let run = false;
57          do Output.printString("PROGRAM FOR READING STRING FROM THE KEYBOARD.");
58          do Output.println();
59          do Output.printString("Please press 'ESC' to move to the next step.");
60          do Output.println();
61           do Output.println();
62          let input = 0;
63
64          while (~run) {
65              let s = Keyboard.readLine("Please enter the string: ");
66              do Output.println();
67              do Output.printString(s);
68              do Output.println();
69              let input = Keyboard.readChar();
70              do Output.println();
71
72           if (input = 140) {
73              do Sys.wait(100);
74              do Output.println();
75              do Output.println();
76              do Output.printString("Moving to read integer from user.");
77              do Sys.wait(1000);
```

```
77              do Sys.wait(1000);
78              do Screen.clearScreen();
79              do Output.moveCursor(0,0);
80              let run = true;
81           }
82          }
83
84          let run = false;
85          do Output.printString("PROGRAM FOR READING INTEGER FROM THE KEYBOARD.");
86          do Output.println();
87          do Output.println();
88
89          while (~run) {
90              do Output.println();
91              let integer = Keyboard.readInt("Please enter the integer: ");
92              do Output.println();
93              do Output.printInt(integer);
94              let input = Keyboard.readChar();
95              do Output.println();
96
97
98           if (input = 140) {
99              do Sys.wait(1000);
100             do Output.println();
101             do Output.println();
102             do Output.printString("Thank you for using this program.");
```

```
95              do Output.println();
96
97
98          if (input = 140) {
99              do Sys.wait(1000);
100             do Output.println();
101             do Output.println();
102             do Output.printString("Thank you for using this program.");
103             let run = true;
104         }
105     }
106
107     return;
108
109     }
110
111 }
112
```

Complied Main code : -

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  JUPYTER

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\HP\Desktop\Amrita\AMRITA APPS\nand2tetris\nand2tetris\tools> .\JackCompiler.bat .\TERM_PROJECT_SEM_2\Main.jack
Compiling "C:\Users\HP\Desktop\Amrita\AMRITA APPS\nand2tetris\nand2tetris\tools\TERM_PROJECT_SEM_2\Main.jack"
```

Main test file: -

- We have a test code in nand2tertris to crosscheck the keyboard library we created.
- Location nand2tetris→projects→12→keyboardtest.

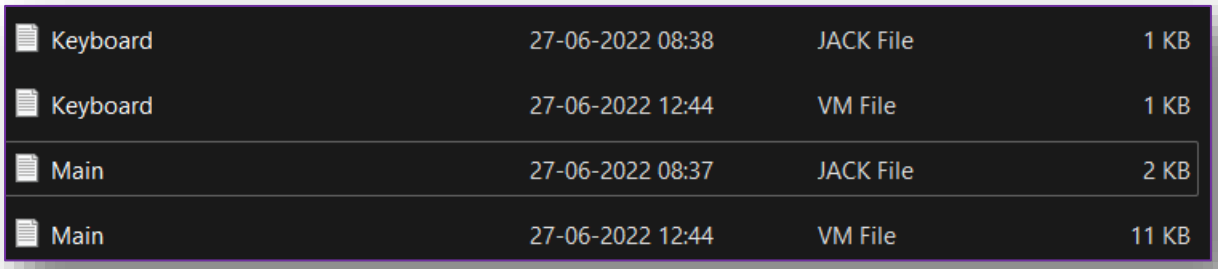
KeyboardTest	07-05-2022 19:54	File folder

```
class Main {

    function void main() {
      var char c, key;
      var String s;
      var int i;
      var boolean ok;

        let ok = false;
        do Output.printString("keyPressed test:");
        do Output.println();
        while (~ok) {
            do Output.printString("Please press the 'Page Down' key");
            while (key = 0) {
                let key = Keyboard.keyPressed();
            }
            let c = key;
            while (~(key = 0)) {
                let key = Keyboard.keyPressed();
            }

            do Output.println();

            if (c = 137) {
                    do Output.printString("ok");
                  do Output.println();
                  let ok = true;
              }
          }

          let ok = false;
        do Output.printString("readChar test:");
        do Output.println();
        do Output.printString("(Verify that the pressed character is echoed to the screen)");
        do Output.println();
        while (~ok) {
            do Output.printString("Please press the number '3': ");
          let c = Keyboard.readChar();
```

```
      do Output.println();

      if (c = 51) {
            do Output.printString("ok");
            do Output.println();
            let ok = true;
        }
    }

let ok = false;
  do Output.printString("readLine test:");
  do Output.println();
  do Output.printString("(Verify echo and usage of 'backspace')");
  do Output.println();
  while (~ok) {
    let s = Keyboard.readLine("Please type 'JACK' and press enter: ");

        if (s.length() = 4) {
            if ((s.charAt(0) = 74) & (s.charAt(1) = 65) & (s.charAt(2) = 67) & (s.charAt(3) = 75)) {
                do Output.printString("ok");
                do Output.println();
                let ok = true;
            }
    }
}

    let ok = false;
  do Output.printString("readInt test:");
  do Output.println();
  do Output.printString("(Verify echo and usage of 'backspace')");
  do Output.println();
  while (~ok) {
        let i = Keyboard.readInt("Please type '-32123' and press enter: ");

        if (i = (-32123)) {
            do Output.printString("ok");
            do Output.println();
            let ok = true;
        }
    }
```

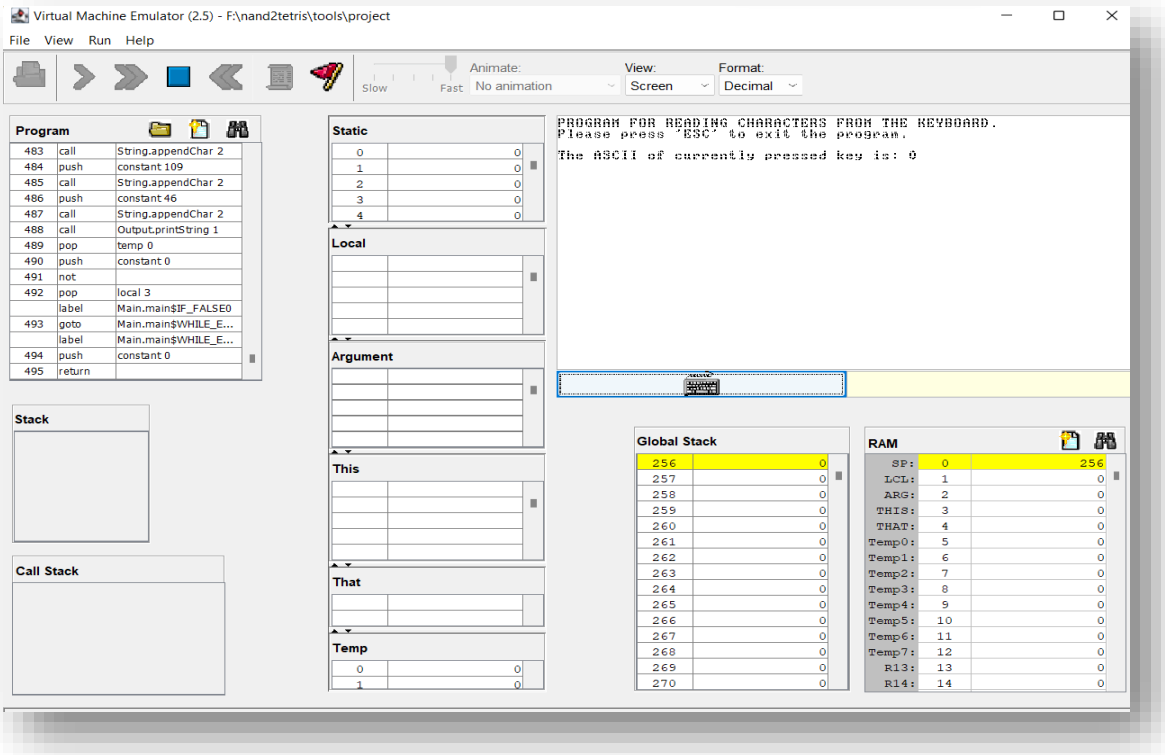
```
        do Output.println();
        do Output.printString("Test completed successfully");

        return;
    }
}
```

Keep all VM files in same folder

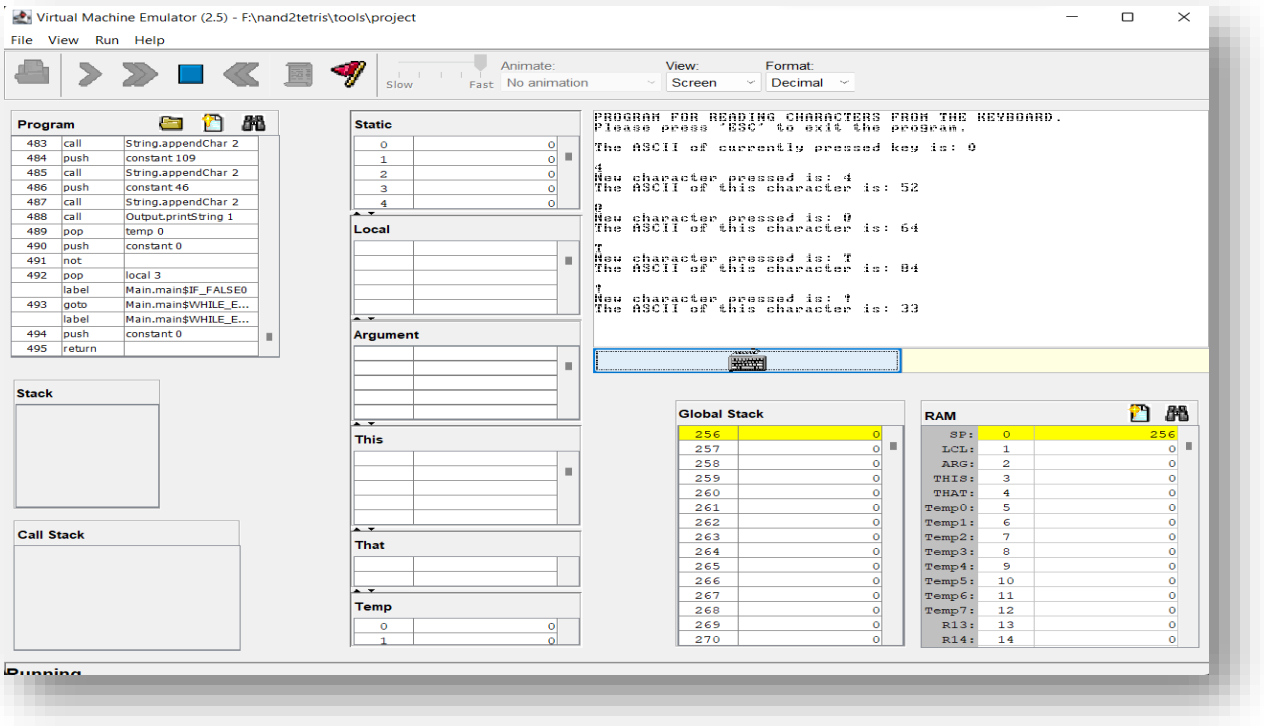


VM Implementation

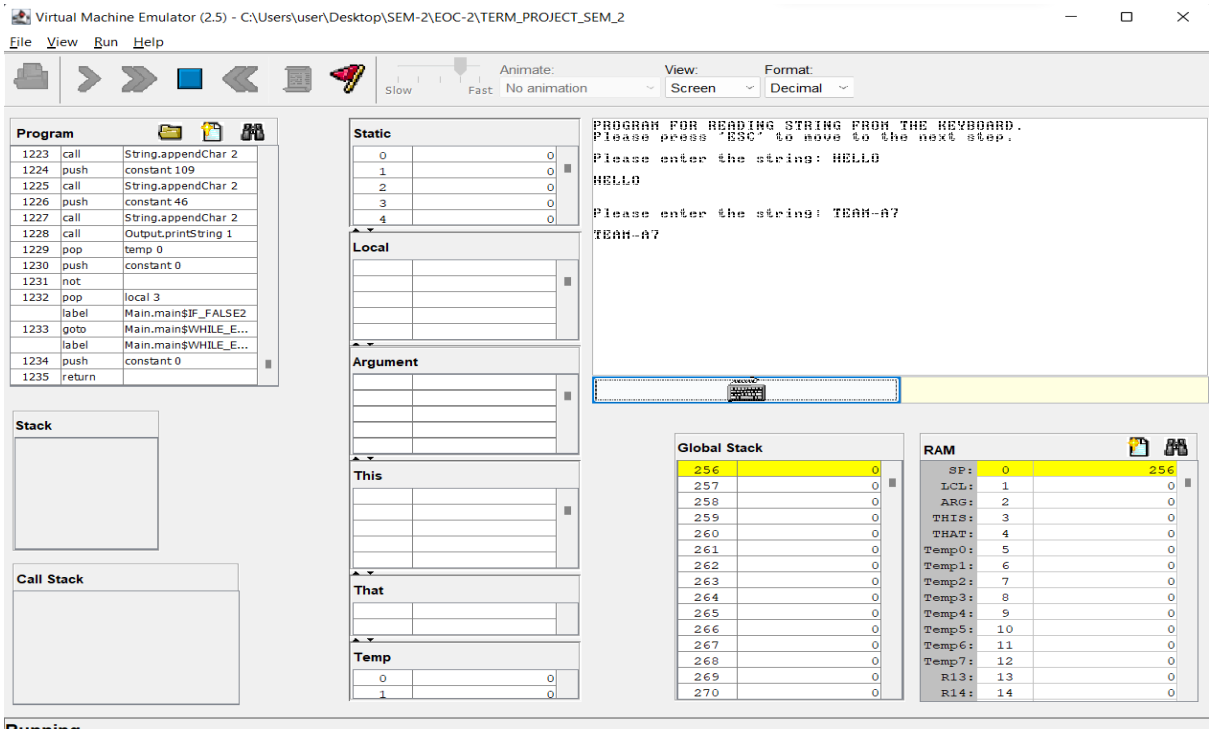
- **Returns the ASCII code (as char) of the currently pressed key, or 0 if no key is currently pressed.**
- **Reads their next character from the keyboard: wait until a key is pressed and then released, then echos the key to the screen and returns the value of the pressed key.**
- **Reads their next character from the keyboard: wait until a key is pressed and then released, then echos the key to the screen and returns the value of the pressed key.**



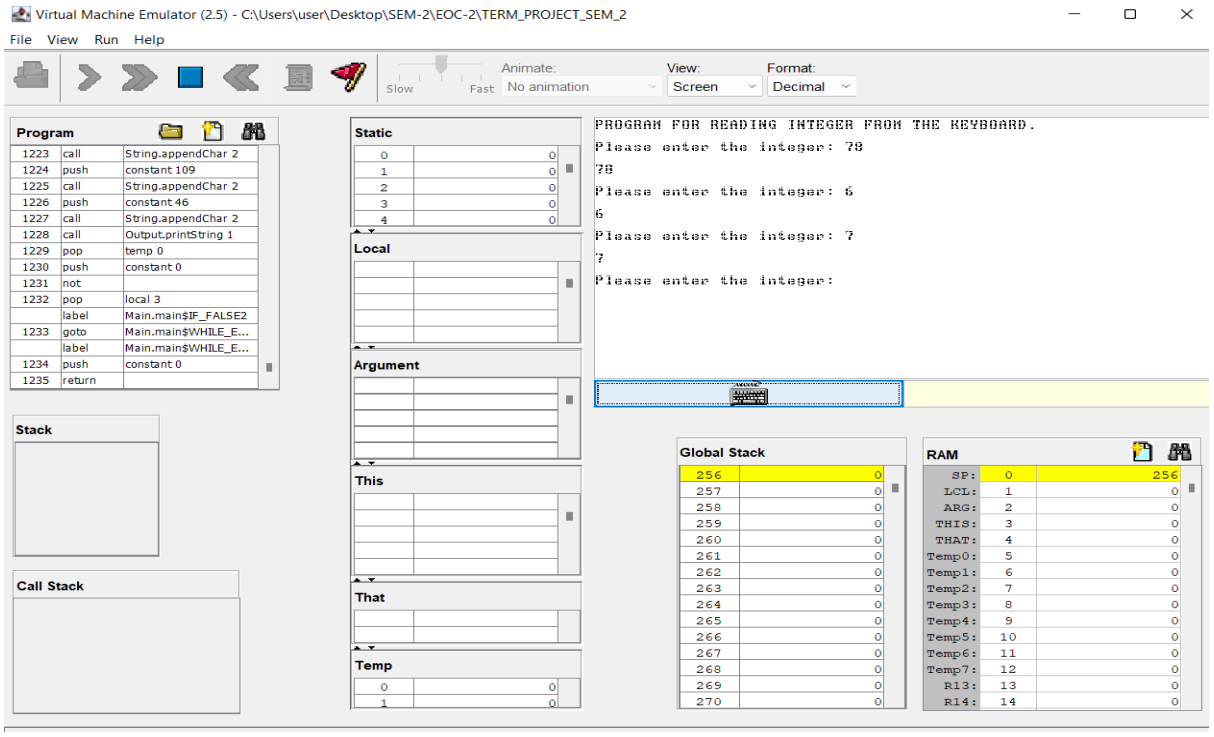
- **Reads their next character from the keyboard: wait until a key is pressed and then released, then echos the key to the screen and returns the value of the pressed key.**



- **Prints the message on the screen, reads the next line from the keyboard ,echoes the line to the screen, and returns it's value. This method handles user backspaces.**



Prints the message on the screen, reads the next line from the keyboard , echoes the line to the screen, and returns the integer until the first non-numeric character in the line. This method handles user backspaces



Part-2

- **Write your own general purpose assembler to convert the assembly program into 16 bit machine codes also use the Nand2tetris assembler for conversion. (Your own assembler should be developed by you and is different from the assembler tool supplied along with the book) .**

ASSEMBLER CODE

Assembler.py

Assembler.py > Assembler > get_hack_file

```python
import sys
import Code
import Parser
import SymbolTable


class Assembler:
    """
    Reads Progam.asm source file and creates a new file Program.hack which has the assembled machine code as a text file.
    The Assembly is implemented as two stages or two passes. The first pass scans the whole program and registers
    symbols in the symbol table. The second pass scans the whole program again substituting the symbols with their
    respective addresses in the symbol table, in addition to generating binary machine code and writing the resulting
    assembled machine code to a new file.
    Usage: python Assembler.py Program.asm
    """
    def __init__(self):
        self.symbol_address = 16
        self.symbols_table = SymbolTable.SymbolTable()

    @staticmethod
    def get_hack_file(asm_file):
        """
        Suggests a file name for the Hack Machine Code source file.
        :param asm_file: Program source code file written in Hack Assembly Language.
        :return: String.
        """
        if asm_file.endswith('.asm'):
            return asm_file.replace('.asm', '.hack')
        else:
            return asm_file + '.hack'
```

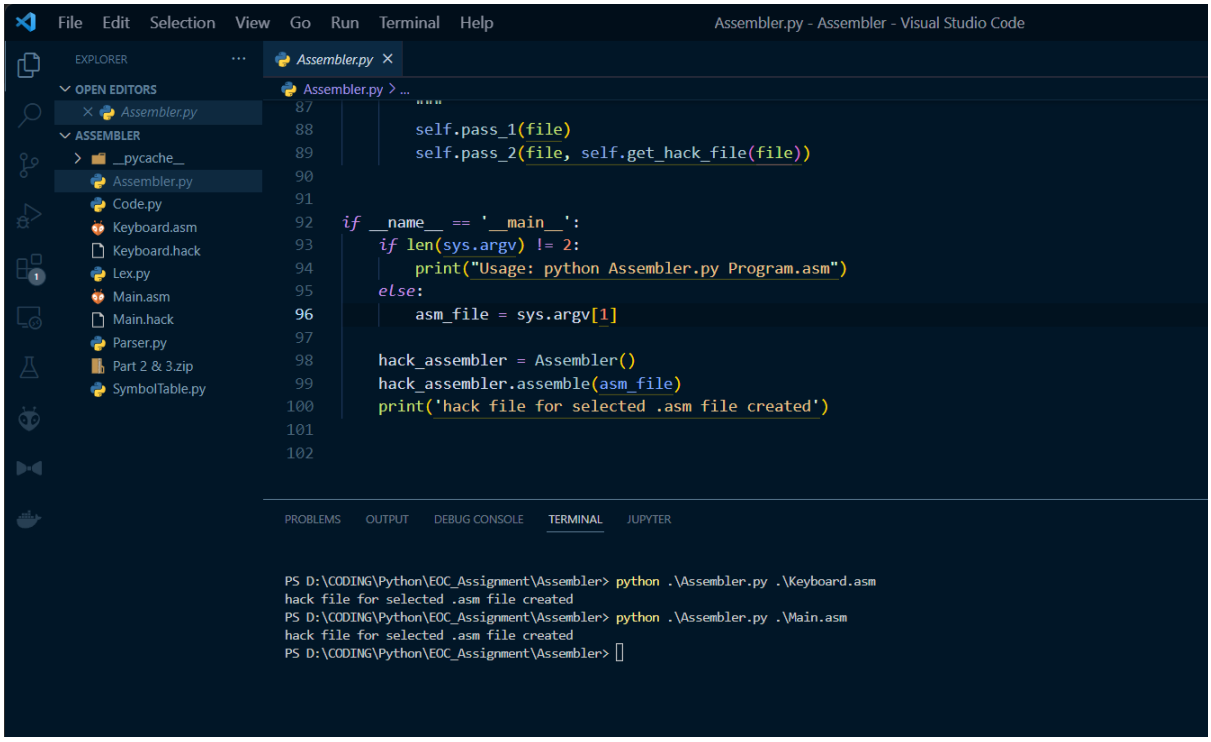
Assembler.py

```python
    def _get_address(self, symbol):
        """
        Helper method. Looks-up the address of a symbol (decimal value, label or variable).
        :param symbol: Symbol or Value.
        :return: Address.
        """
        if symbol.isdigit():
            return symbol
        else:
            if not self.symbols_table.contains(symbol):
                self.symbols_table.add_entry(symbol, self.symbol_address)
                self.symbol_address += 1
            return self.symbols_table.get_address(symbol)

    def pass_1(self, file):
        """
        First compilation pass: Determine memory locations of label definitions: (LABEL).
        :param file:
        :return:
        """
        parser = Parser.Parser(file)
        curr_address = 0
        while parser.has_more_instructions():
            parser.advance()
            inst_type = parser.instruction_type
            if inst_type in [parser.A_INSTRUCTION, parser.C_INSTRUCTION]:
                curr_address += 1
            elif inst_type == parser.L_INSTRUCTION:
                self.symbols_table.add_entry(parser.symbol, curr_address)
```

Assembler.py

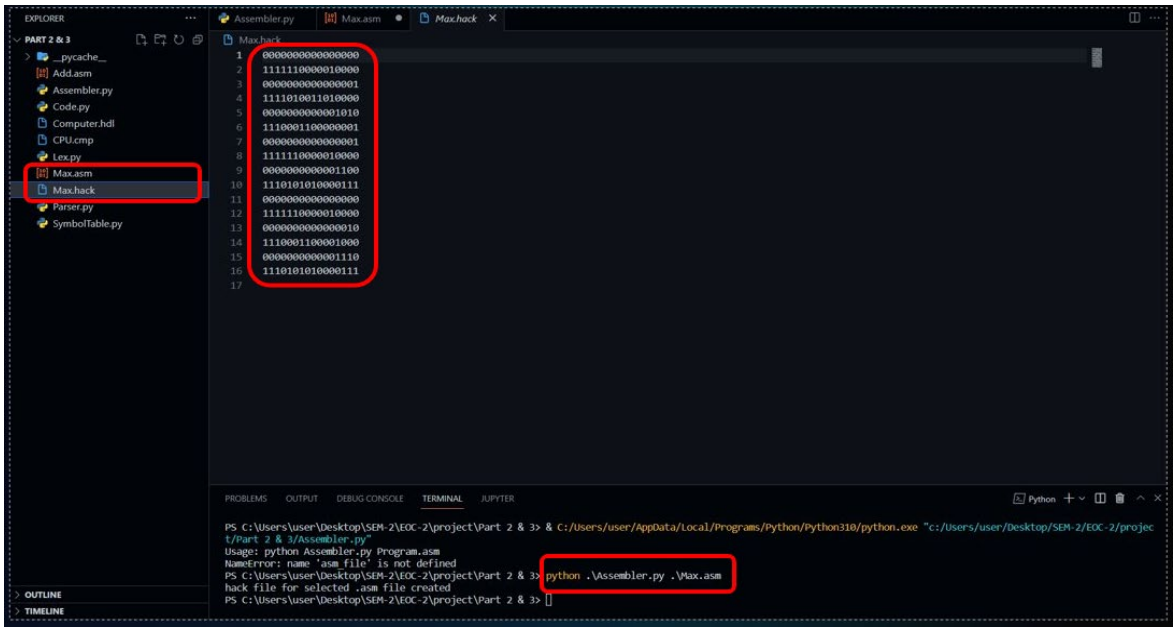
```python
    def pass_2(self, asm_file, hack_file):
        """
        Second compilation pass: Generate hack machine code and write results to output file.
        :param asm_file: The program source code file, written in Hack Asembly Language.
        :param hack_file: Output file to write Hack Machine Code output to.
        :return: None.
        """
        parser = Parser.Parser(asm_file)
        with open(hack_file, 'w', encoding='utf-8') as hack_file:
            code = Code.Code()
            while parser.has_more_instructions():
                parser.advance()
                inst_type = parser.instruction_type
                if inst_type == parser.A_INSTRUCTION:
                    hack_file.write(code.gen_a_instruction(self._get_address(parser.symbol)) + '\n')
                elif inst_type == parser.C_INSTRUCTION:
                    hack_file.write(code.gen_c_instruction(parser.dest, parser.comp, parser.jmp) + '\n')
                elif inst_type == parser.L_INSTRUCTION:
                    pass

    def assemble(self, file):
        """
        The main method. Drives the assembly process.
        :param file: Program source code file, written in the Hack Assembly Language.
        :return: None.
        """
        self.pass_1(file)
        self.pass_2(file, self.get_hack_file(file))
```

```
81
82      def assemble(self, file):
83          """
84          The main method. Drives the assembly process.
85          :param file: Program source code file, written in the Hack Assembly Language.
86          :return: None.
87          """
88          self.pass_1(file)
89          self.pass_2(file, self.get_hack_file(file))
90
91
92  if __name__ == '__main__':
93      if len(sys.argv) != 2:
94          print("Usage: python Assembler.py Program.asm")
95      else:
96          asm_file = sys.argv[1]
97
98      hack_assembler = Assembler()
99      hack_assembler.assemble(asm_file)
100     print('hack file for selected .asm file created')
101
102
```



- Here we created our own assembler to convert .asm files to .hack files.
- We need to keep the files in the same folder in order to convert .asm files to .hack files.

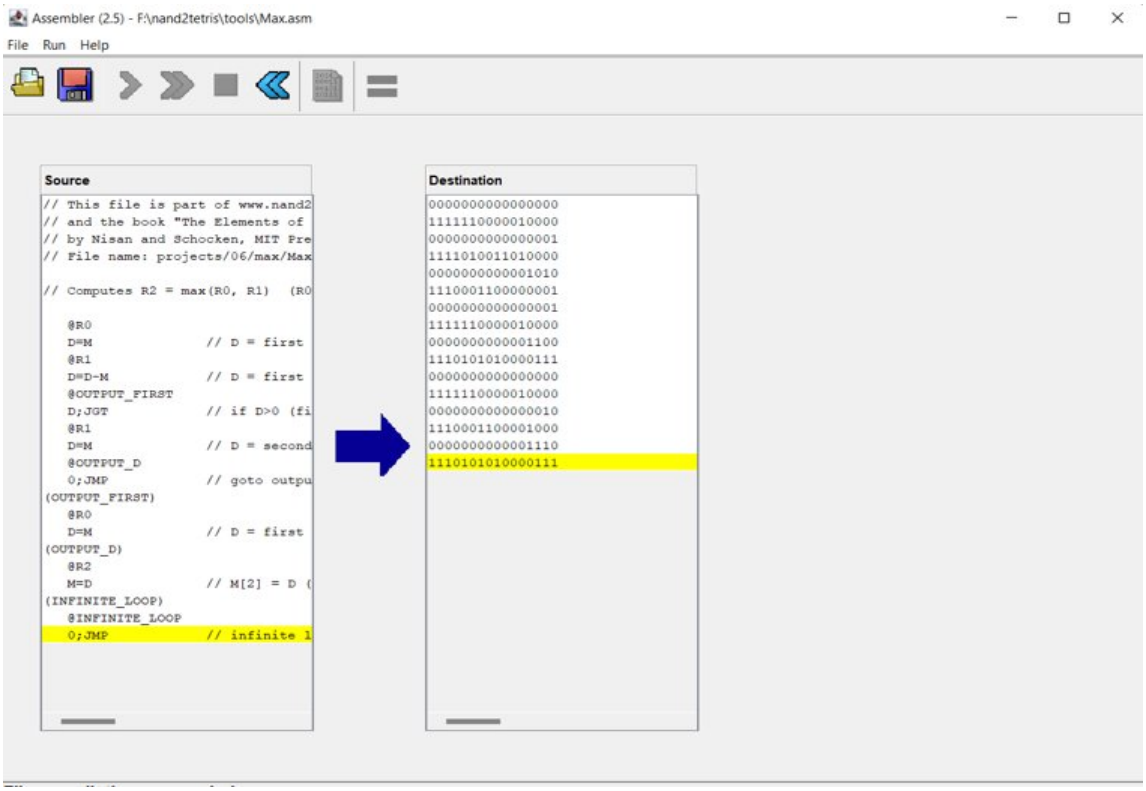
Now in the terminal we use the command "python .\Assembler.py .\Max.asm" to convert asm.file to hack.file.



- After doing the above process we can see a new file in the folder named as “Main.hack” which is a hack file.

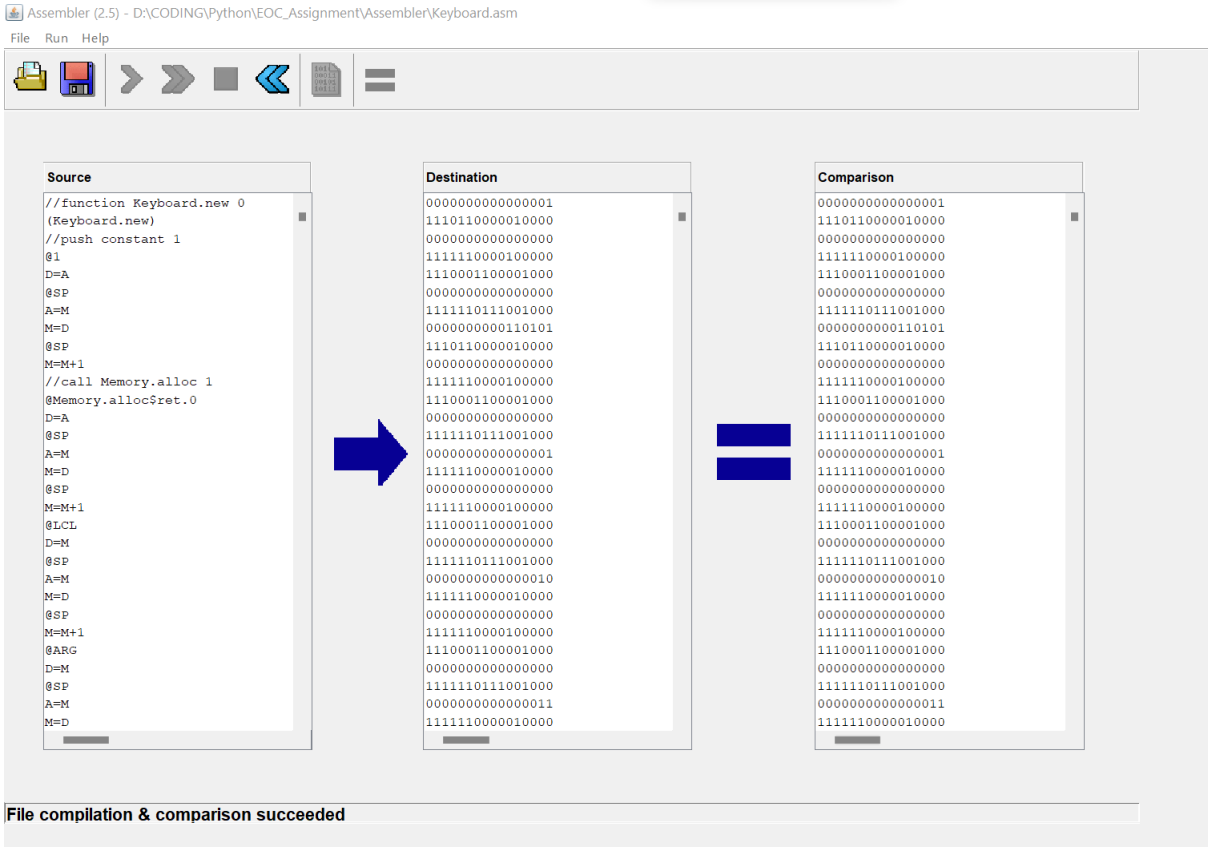
- Now we are going to use assembler tool in nand2tetris to do the same process of converting asm file to hack file.

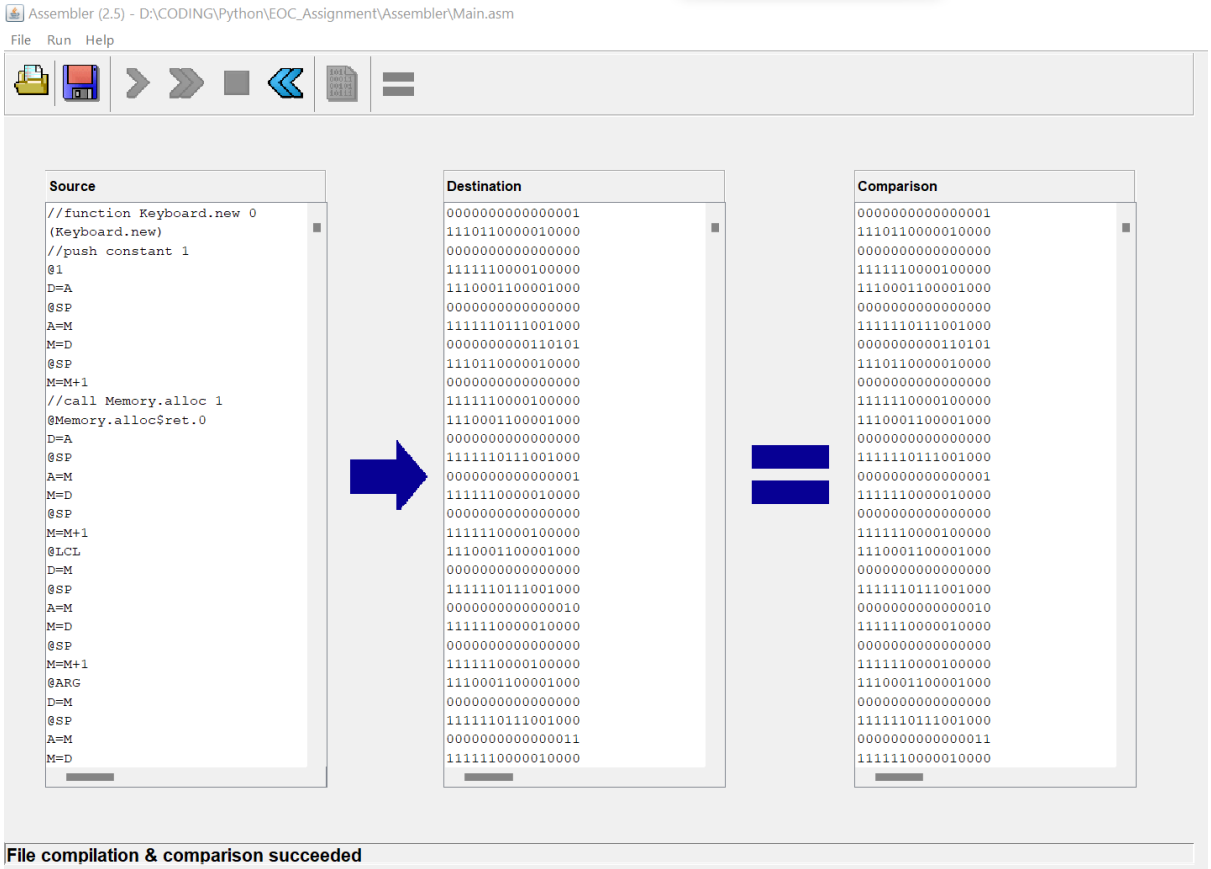
- Open the assembler and load the asm file and then click on the run button to convert it into hack language, then save the file.



- If we equate both the hack files that we obtained earlier using own assembler and nand2tertis are identical.

COMPARISON PHASE





Part-3

Dump the machine codes generated by your own assembler into the RAM.hdl. Then design the architecture for implementing your machine instructions to get the result. (Here I presume that you already have the hdl scripts of all the hardware's used by the CPU. Additional thing what you have to do is connecting the address and data buses during the execution).

- **What is meant by 16 bit instructions?**

1. There are two sorts of 16-bit instructions: A instructions and C instructions. Now, from the 0th to the 15th bit, if the first bit (i.e., the T-pin) is zero, it is an A instruction; if the first three bits are 1 1 1, or if the addition of the first bit is certain, it is a C instruction.
2. If the first bit is zero, the instruction is an A; if the first bit is one, the instruction is a C-instruction.
3. For the A instruction, the remaining 15 bits are value, which is saved in the A register and then outputs the value.
4. However, if I have a C instruction, then everything is entirely different. In this case, the 13 instruction bits are decoded into the Op-code (treat it as letter "a"), ALU control bits, Destination load bits, and Jump bits.

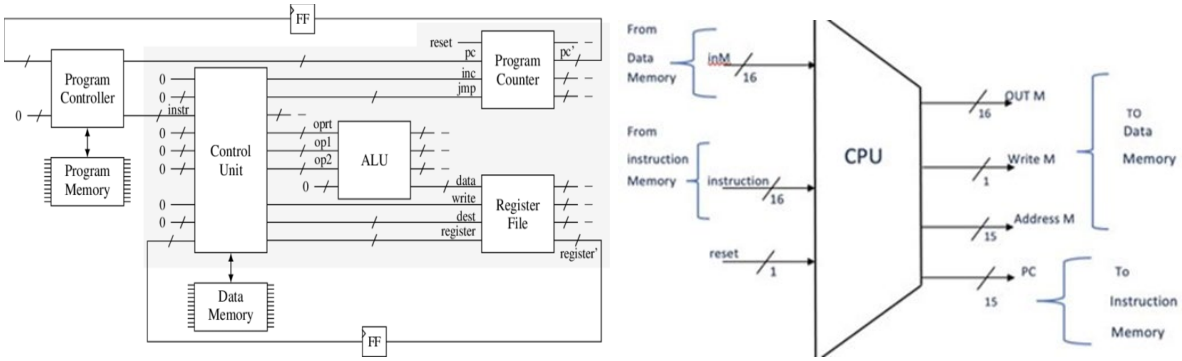
Computer hdl file

```
/**
 * The HACK computer, including CPU, ROM and RAM.
 * When reset is 0, the program stored in the computer's ROM executes.
 * When reset is 1, the execution of the program restarts.
 * Thus, to start a program's execution, reset must be pushed "up" (1)
 * and "down" (0). From this point onward the user is at the mercy of
 * the software. In particular, depending on the program's code, the
 * screen may show some output and the user may be able to interact
 * with the computer via the keyboard.
 */

CHIP Computer {
    IN reset;

    PARTS:
    ROM32K(address=pc, out=instruction);
    CPU(inM=inM, instruction=instruction, reset=reset, outM=outM, writeM=writeM, addressM=addressM, pc=pc);
    Memory(in=outM, load=writeM, address=addressM, out=inM);
}
```

Central Processing Unit(C.P.U)

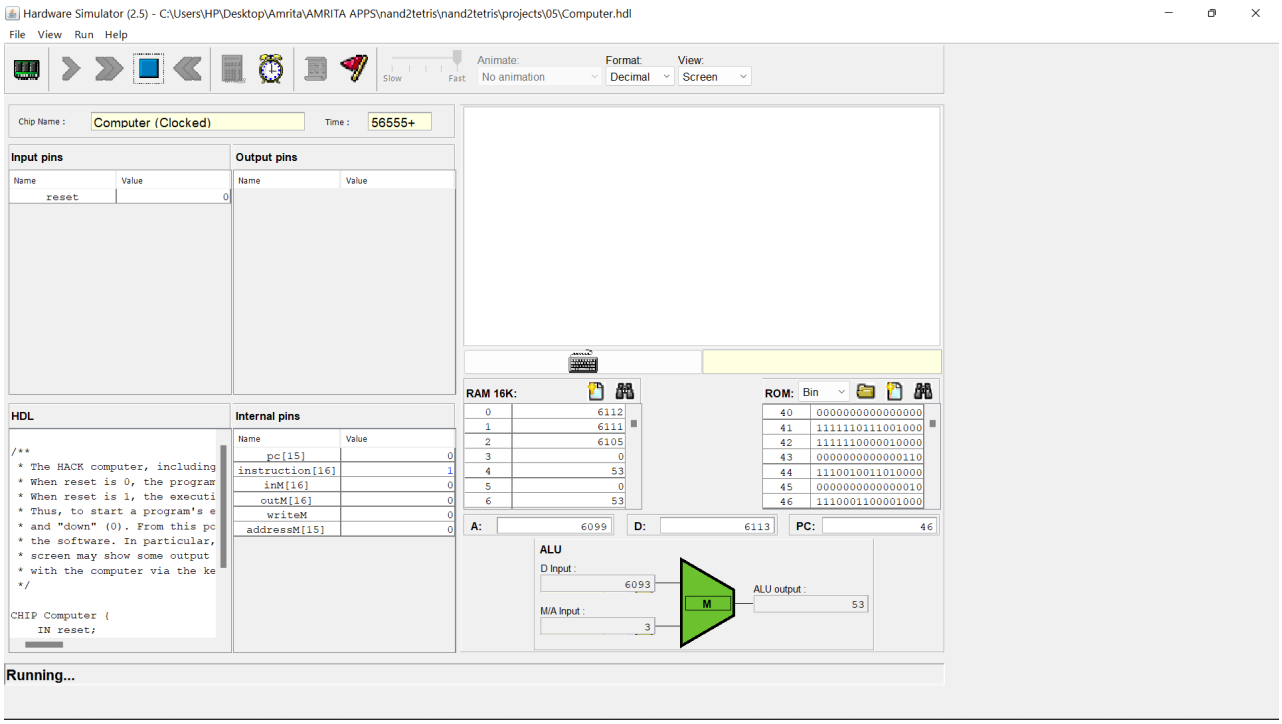
- The control unit is in charge of all CPU functions, including ALU operations, data transfer inside the CPU, and the interchange of data and control signals across external interfaces.
- The control unit is in charge of all CPU functions, including ALU operations, data transfer inside the CPU, and the interchange of data and control signals across external interfaces.

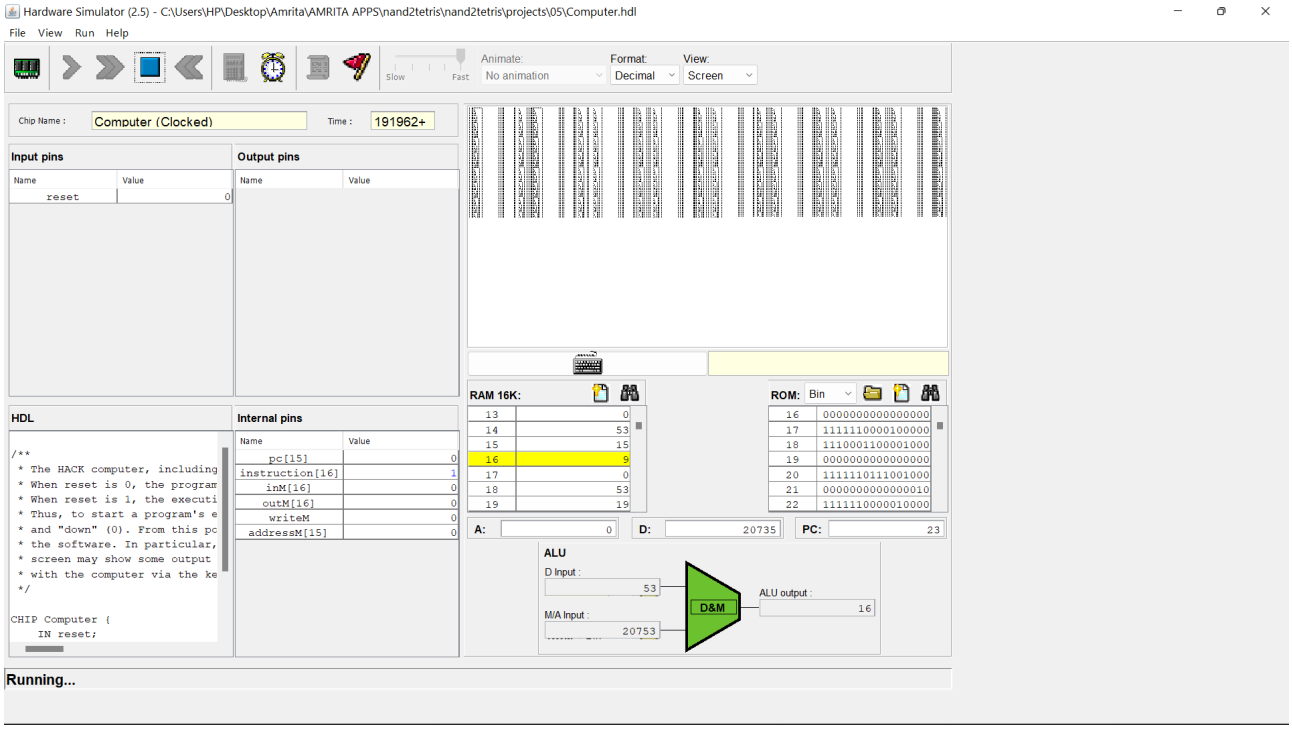
CPU DESIGN

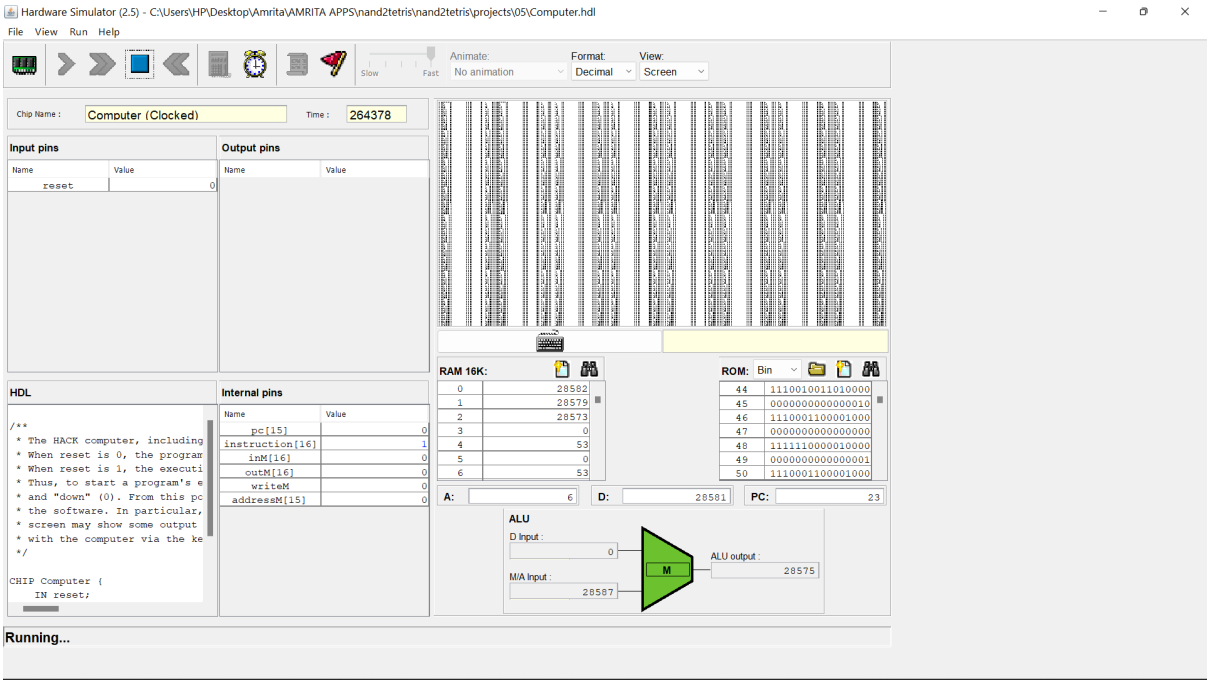


- **Instruction Memory -**It contains a 16 bit 'instruction' as input and the output of PC can be explained based on Instruction memory.

Working with .hack file of KEYBOARD & MAIN



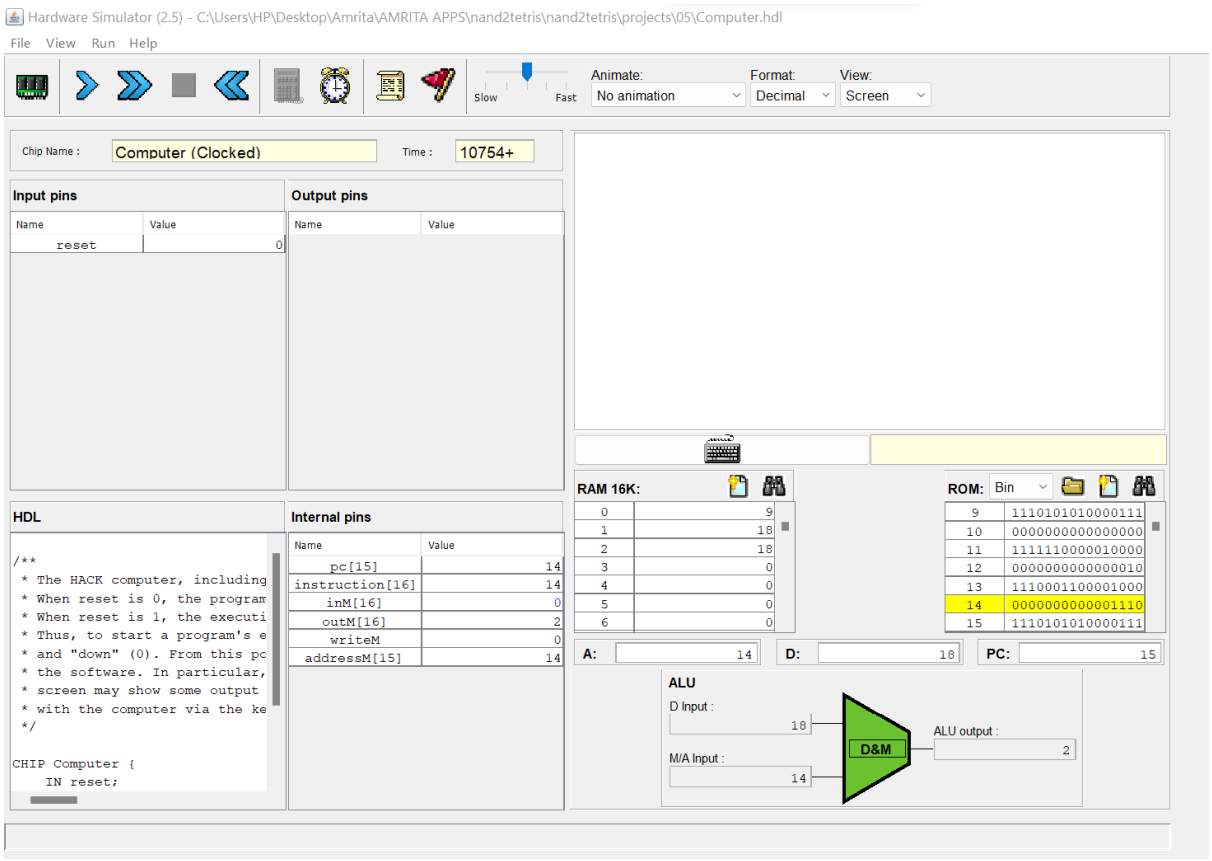




Error

When we load the code on hardware simulator and run that code we can see that code will never stop, will continue in an infinite loop but will never give desired output.
The reason behind this error is the information transfer and working of nand2tetris package as a whole, ASCII characters, calling of different files is not possible in terms of real time working, even though code gets converted into hack from .jack file.
So we are going to use a simple alternate code of MAX which will compare two inputs and displays the maximum number out of the two input.

ALTERNATE CODE[MAX HACK]



EXTRAS

Keyboard.jack TERM_TEST | Keyboard.jack TERM_PROJECT_SEM_2 | Main.jack | String.jack

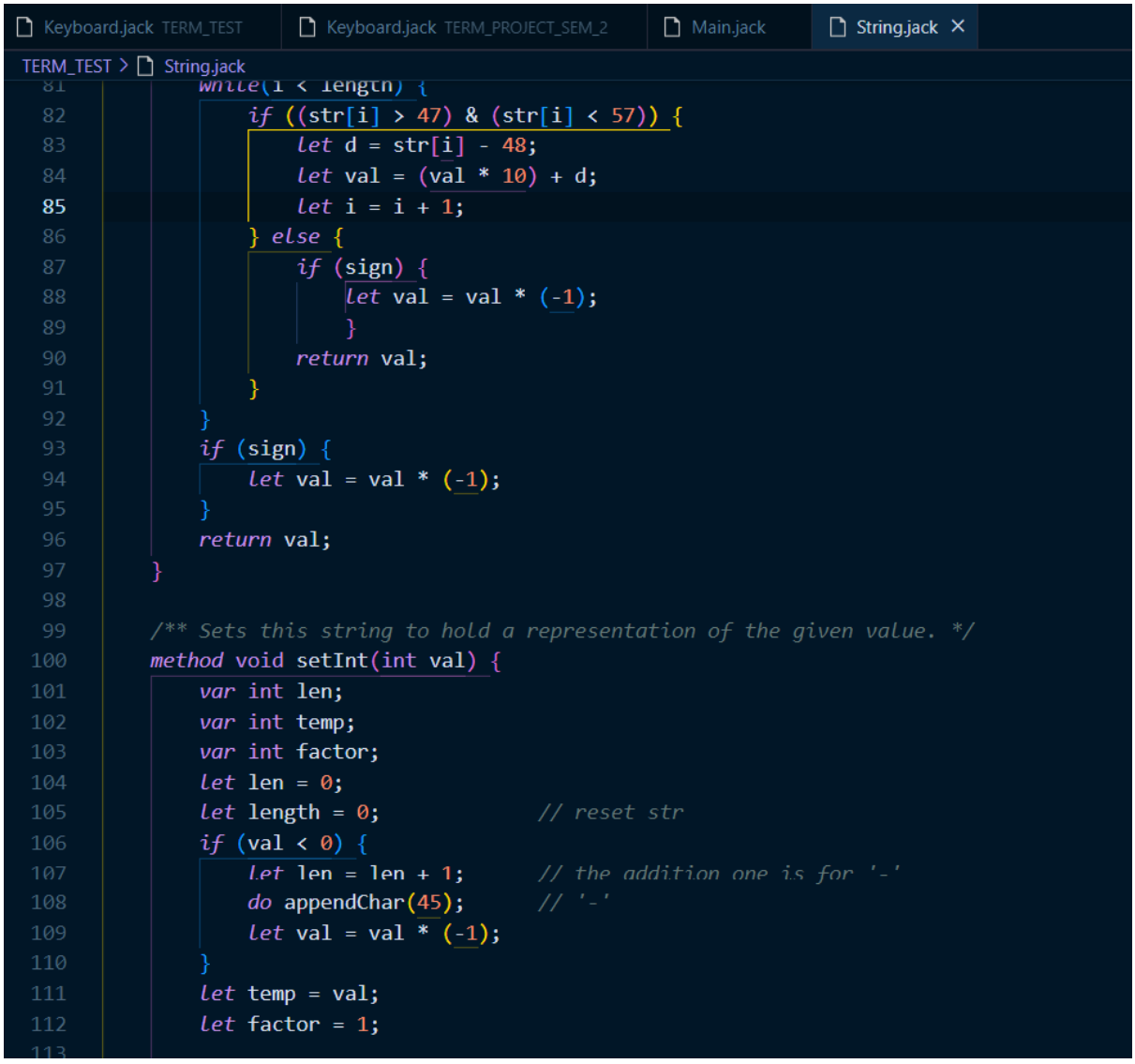
TERM_TEST > String.jack

```
1   // This file is part of www.nand2tetris.org
2   // and the book "The Elements of Computing Systems"
3   // by Nisan and Schocken, MIT Press.
4   // File name: projects/12/String.jack
5
6   /**
7    * Represents character strings. In addition for constructing and disposing
8    * strings, the class features methods for getting and setting individual
9    * characters of the string, for erasing the string's last character,
10   * for appending a character to the string's end, and more typical
11   * string-oriented operations.
12   */
13  class String {
14      field Array str;
15      field int length;
16      field int capacity;
17
18      /** constructs a new empty string with a maximum length of maxLength
19       *  and initial length of 0. */
20      constructor String new(int maxLength) {
21          if (~(maxLength = 0)) {
22              let str = Array.new(maxLength);
23          } else {        // zero-capacity string
24              let str = Array.new(1);
25          }
26          let length = 0;
27          let capacity = maxLength;
28          return this;
29      }
```


Keyboard.jack TERM_TEST
Keyboard.jack TERM_PROJECT_SEM_2
Main.jack
String.jack
TERM_TEST > String.jack
/** Disposes this string. */
method void dispose() {
do Memory.deAlloc(str);
return;
}
/** Returns the current length of this string. */
method int length() {
return length;
}
/** Returns the character at the j-th location of this string. */
method char charAt(int j) {
return str[j];
}
/** Sets the character at the j-th location of this string to c. */
method void setCharAt(int j, char c) {
let str[j] = c;
return;
}
/** Appends c to this string's end and returns this string. */
method String appendChar(char c) {
let str[length] = c;
let length = length + 1;
return this;
}

TERM_TEST > String.jack

```
59
60      /** Erases the last character from this string. */
61      method void eraseLastChar() {
62          let str[length-1] = 0;
63          let length = length - 1;
64          return;
65      }
66
67      /** Returns the integer value of this string,
68       *  until a non-digit character is detected. */
69      method int intValue() {
70          var int val;
71          var int i;
72          var int d;
73          var bool sign;
74          let val = 0;
75          let i = 0;
76          let sign = false;
77          if (str[i] = 45) {
78              let sign = true;
79              let i = i + 1;
80          }
81          while(i < length) {
82              if ((str[i] > 47) & (str[i] < 57)) {
83                  let d = str[i] - 48;
84                  let val = (val * 10) + d;
85                  let i = i + 1;
86              } else {
87                  if (sign) {
88                      let val = val * (-1);
89                      }
90                  return val;
91              }
92          }
```


Keyboard.jack TERM_TEST
Keyboard.jack TERM_PROJECT_SEM_2
Main.jack
String.jack
TERM_TEST > String.jack
while(i < length) {
if ((str[i] > 47) & (str[i] < 57)) {
let d = str[i] - 48;
let val = (val * 10) + d;
let i = i + 1;
} else {
if (sign) {
let val = val * (-1);
}
return val;
}
}
if (sign) {
let val = val * (-1);
}
return val;
}
/** Sets this string to hold a representation of the given value. */
method void setInt(int val) {
var int len;
var int temp;
var int factor;
let len = 0;
let length = 0;
// reset str
if (val < 0) {
let len = len + 1;
// the addition one is for '-'
do appendChar(45);
// '-'
let val = val * (-1);
}
let temp = val;
let factor = 1;


Keyboard.jack TERM_TEST
Keyboard.jack TERM_PROJECT_SEM_2
Main.jack
String.jack
TERM_TEST > String.jack
112 let factor = 1;
113
114 // count length
115 while((temp / 10) > 0) {
116 let len = len + 1;
117 let temp = temp / 10;
118 let factor = factor * 10;
119 }
120 let len = len + 1; // integer has at least one digit
121
122 // set string
123 while(factor > 0) {
124 do appendChar((val/factor)+48);
125 let val = val - (factor * (val / factor));
126 let factor = factor / 10;
127 }
128
129 return;
130 }
131
132 /** Returns the new line character. */
133 function char newLine() {
134 return 128;
135 }
136
137 /** Returns the backspace character. */
138 function char backSpace() {
139 return 129;
140 }
141
142 /** Returns the double quote (") character. */
143 function char doubleQuote() {
144 return 34;
145 }

Conclusion

From this project we have learned working on JACK language and how JACK language can be used to create IN-BUILT library(keyboard) within a time frame with much more functionalities, without any prime limitations.
Working of assembler is understandable on a whole new level where one can simply create a nand2tetris software with basic knowledge of HIGH-LEVEL programming language.
Generating code in BITS and working with 16-BIT code is done on much easier level which can be tested using software available in nand2tetris folder.