ELEMENTS OF COMPUTING

.END SEMESTER PROJECT

TEAM MEMBERS

01 G.SRI VATSANKA
CB.EN.U4AIE.21010

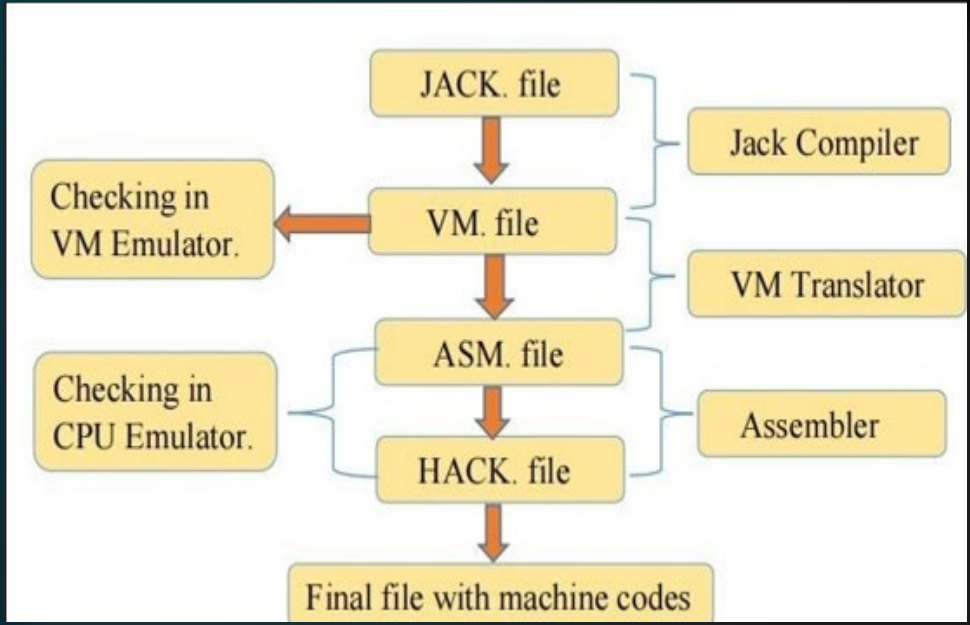
02 G.HIMAMSH
CB.EN.U4AIE.21014

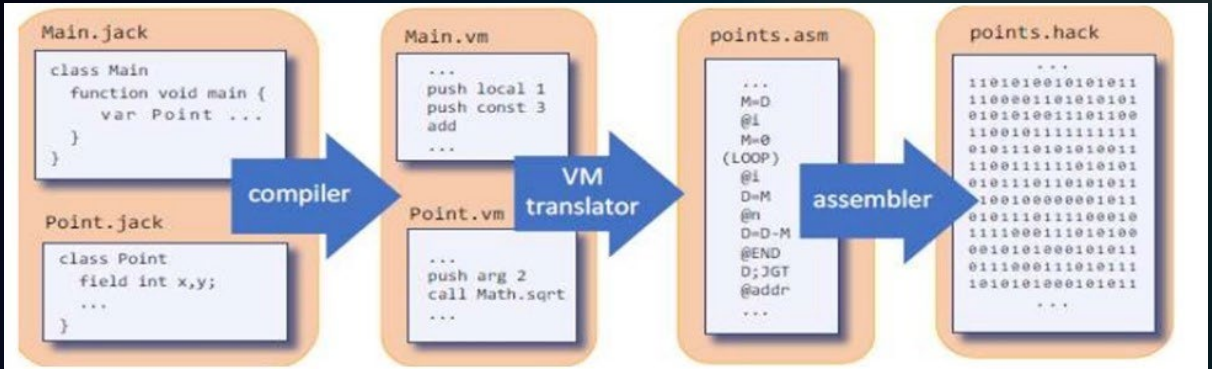
03 M.PRASANNA TEJA
CB.EN.U4AIE.21035

04 VIKHYAT BANSAL
CB.EN.U4AIE.21076

PART-1

Conversion of files





Required Steps:

Step-1

- We will convert .Jack files to .VM files.

Step-2

- Conversion of .VM files to .ASM files.

Step-3

- .ASM files to hack files conversion.

INTRODUCTION

- 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. The NAND2TETRIS folder is the source of this program.

JACK LANGUAGE

- Jack is simple Object -based language that can be used to write High-Level programs .It has been a basic features and flavor of modern object-oriented language like java with much simpler syntax and no support for inheritance.

- We use jack language as a means for teaching of,

1. Keyboard class allows the reading from a standard keyboard.
2. How to create a compiler.
3. To see how compiler and language interfere with the program.

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.

THE PROJECT

Implementing 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. Reads the 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.
3. 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. 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.

Keyboard Class

- 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.

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;
    }
```

```

    function String readLine(String text) {

        var String val;
        var char c;

        let val =String.new(69);
        let c = 0;

        do Output.printString(text);

        while (~(c=128)) {
            let c = Keyboard.readChar();
            if (c=129){
                do val.eraseLastChar();
                do Output.printChar(129);
            }
            else{
                do Output.printChar(c);
                do val.appendChar(c);
            }
        }
        do val.eraseLastChar();
        return val;
    }
```

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;
    }
```

```

    function String readLine(String text) {

        var String val;
        var char c;

        let val =String.new(69);
        let c = 0;

        do Output.printString(text);

        while (~(c=128)) {
            let c = Keyboard.readChar();
            if (c=129){
                do val.eraseLastChar();
                do Output.printChar(129);
            }
            else{
                do Output.printChar(c);
                do val.appendChar(c);
            }
        }
        do val.eraseLastChar();
        return val;
    }
```

Keyboard jack code

```
    function int readInt(String message) {

        var String val;
        var Int ival;
        var char c;

        let val =String.new(69);
        let c = 0;

        do Output.printString(message);

        while (~(c=128)) {
            let c = Keyboard.readChar();
            if (c=129){
                do val.eraseLastChar();
                do Output.printChar(c);

            }
            else{
                do val.appendChar(c);
                do Output.printChar(c);
            }
        }
        do val.eraseLastChar();
        let ival=val.intValue();
        return ival;
    }

}
```

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>
```

Keyboard.vm code

```
function Keyboard.new 0
push constant 1
call Memory.alloc 1
pop pointer 0
push pointer 0
return
function Keyboard.init 0
push constant 24576
pop static 0
push constant 0
return
function Keyboard.KeyPressed 0
push constant 0
push static 0
add
pop pointer 1
push that 0
return
function Keyboard.readChar 1
label WHILE_EXP0
call Keyboard.KeyPressed 0
push constant 0
eq
not
if-goto WHILE_END0
goto WHILE_EXP0
label WHILE_END0
call Keyboard.KeyPressed 0
pop local 0
label WHILE_EXP1
call Keyboard.KeyPressed 0
push constant 0
eq
not
not
if-goto WHILE_END1
goto WHILE_EXP1
label WHILE_END1
push local 0
return
function Keyboard.readLine 2
push constant 69
call String.new 1
pop local 0
push constant 0
pop local 1
push argument 0
call Output.printString 1
pop temp 0
label WHILE_EXP0
push local 1
push constant 128
eq
not
not
if-goto WHILE_END0
call Keyboard.readChar 0
pop local 1
push local 1
push constant 129
eq
if-goto IF_TRUE0
goto IF_FALSE0
label IF_TRUE0
push local 0
call String.eraseLastChar 1
pop temp 0
push constant 129
call Output.printChar 1
pop temp 0
goto IF_END0
label IF_FALSE0
push local 1
call Output.printChar 1
pop temp 0
push local 0
push local 1
call String.appendChar 2
pop temp 0
label IF_END0
goto WHILE_EXP0
label WHILE_END0
push local 0
call String.eraseLastChar 1
pop temp 0
push local 0
return
function Keyboard.readInt 3
push constant 69
call String.new 1
pop local 0
push constant 0
pop local 2
push argument 0
call Output.printString 1
pop temp 0
label WHILE_EXP0
push local 2
push constant 128
eq
not
not
if-goto WHILE_END0
call Keyboard.readChar 0
pop local 2
push local 2
push constant 129
eq
if-goto IF_TRUE0
goto IF_FALSE0
label IF_TRUE0
push local 0
call String.eraseLastChar 1
pop temp 0
push local 2
call Output.printChar 1
pop temp 0
goto IF_END0
label IF_FALSE0
push local 0
push local 2
call String.appendChar 2
pop temp 0
push local 2
call Output.printChar 1
pop temp 0
label IF_END0
goto WHILE_EXP0
label WHILE_END0
push local 0
call String.eraseLastChar 1
pop temp 0
push local 0
call String.intValue 1
pop local 1
push local 1
return
```

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

```
        }

    }

        let run = false;
        do Output.printString("PROGRAM FOR READING STRING FROM THE KEYBOARD.");
        do Output.println();
        do Output.printString("Please press 'ESC' to move to the next step.");
        do Output.println();
         do Output.println();
        let input = 0;

        while (~run) {
            let s = Keyboard.readLine("Please enter the string: ");
            do Output.println();
            do Output.printString(s);
            do Output.println();
            let input = Keyboard.readChar();
            do Output.println();

         if (input = 140) {
            do Sys.wait(100);
            do Output.println();
            do Output.println();
            do Output.printString("Moving to read integer from user.");
            do Sys.wait(1000);
```

```
        do Sys.wait(1000);
        do Screen.clearScreen();
        do Output.moveCursor(0,0);
        let run = true;
    }
}

let run = false;
do Output.printString("PROGRAM FOR READING INTEGER FROM THE KEYBOARD.");
do Output.println();
do Output.println();

while (~run) {
    do Output.println();
    let integer = Keyboard.readInt("Please enter the integer: ");
    do Output.println();
    do Output.printInt(integer);
    let input = Keyboard.readChar();
    do Output.println();


    if (input = 140) {
        do Sys.wait(1000);
        do Output.println();
        do Output.println();
        do Output.printString("Thank you for using this program.");
```

Compiled MAIN code

```
            do Output.println();


        if (input = 140) {
            do Sys.wait(1000);
            do Output.println();
            do Output.println();
            do Output.printString("Thank you for using this program.");
            let run = true;
        }
    }

    return;

    }

}
```

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.vm code

```
function Main.main 7
push constant 0
pop local 3
push constant 49
call String.new 1
push constant 80
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 79
call String.appendChar 2
push constant 71
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 65
call String.appendChar 2
push constant 77
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 70
call String.appendChar 2
push constant 79
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 69
call String.appendChar 2
push constant 65
call String.appendChar 2
push constant 68
call String.appendChar 2
push constant 73
call String.appendChar 2
push constant 78
call String.appendChar 2
push constant 71
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 67
call String.appendChar 2
push constant 72
call String.appendChar 2
push constant 65
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 65
call String.appendChar 2
push constant 67
call String.appendChar 2
push constant 84
call String.appendChar 2
push constant 69
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 83
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 70
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 79
call String.appendChar 2
push constant 77
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 84
call String.appendChar 2
push constant 72
call String.appendChar 2
push constant 69
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 75
call String.appendChar 2
push constant 69
call String.appendChar 2
push constant 89
call String.appendChar 2
push constant 66
call String.appendChar 2
push constant 79
call String.appendChar 2
push constant 65
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 68
call String.appendChar 2
push constant 46
call String.appendChar 2
call Output.printString 1
pop temp 0
call Output.println 0
pop temp 0
push constant 44
call String.new 1
push constant 80
call String.appendChar 2
push constant 108
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 97
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 112
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 39
call String.appendChar 2
push constant 69
call String.appendChar 2
push constant 83
call String.appendChar 2
push constant 67
call String.appendChar 2
push constant 39
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 111
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 109
call String.appendChar 2
push constant 111
call String.appendChar 2
push constant 118
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 111
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 104
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 110
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 120
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 112
call String.appendChar 2
push constant 46
call String.appendChar 2
call Output.printString 1
pop temp 0
call Output.println 0
pop temp 0
call Output.println 0
pop temp 0
push constant 31
call String.new 1
push constant 75
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 112
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 112
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 105
call String.appendChar 2
push constant 110
call String.appendChar 2
push constant 103
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 97
call String.appendChar 2
push constant 110
call String.appendChar 2
push constant 121
call String.appendChar 2
```

```
push constant 32
call String.appendChar 2
push constant 107
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 121
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 111
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 97
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 46
call String.appendChar 2
call Output.printString 1
pop temp 0
call Output.println 0
pop temp 0
call Output.println 0
pop temp 0
call Keyboard.KeyPressed 0
pop local 0
push constant 1000
call Sys.wait 1
pop temp 0
push constant 39
call String.new 1
push constant 84
call String.appendChar 2
push constant 104
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 65
call String.appendChar 2
push constant 83
call String.appendChar 2
push constant 67
call String.appendChar 2
push constant 73
call String.appendChar 2
push constant 73
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 111
call String.appendChar 2
push constant 102
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 99
call String.appendChar 2
push constant 117
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 110
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 108
call String.appendChar 2
push constant 121
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 112
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 100
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 107
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 121
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 105
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 58
call String.appendChar 2
push constant 32
call String.appendChar 2
call Output.printString 1
pop temp 0
push local 0
call Output.printInt 1
pop temp 0
call Output.println 0
pop temp 0
label WHILE_EXP0
push local 3
not
not
if-goto WHILE_END0
push constant 500
call Sys.wait 1
pop temp 0
call Output.println 0
pop temp 0
call Keyboard.readChar 0
pop local 1
call Output.println 0
pop temp 0
push constant 100
call Sys.wait 1
pop temp 0
push constant 26
call String.new 1
push constant 78
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 119
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 99
call String.appendChar 2
push constant 104
call String.appendChar 2
push constant 97
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 97
call String.appendChar 2
push constant 99
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 112
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 100
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 105
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 58
call String.appendChar 2
push constant 32
call String.appendChar 2
call Output.printString 1
pop temp 0
push local 1
call Output.printChar 1
pop temp 0
call Output.println 0
pop temp 0
push constant 100
call Sys.wait 1
pop temp 0
push constant 32
call String.new 1
push constant 84
call String.appendChar 2
push constant 104
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 65
call String.appendChar 2
push constant 83
call String.appendChar 2
push constant 67
call String.appendChar 2
push constant 73
call String.appendChar 2
push constant 73
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 111
call String.appendChar 2
push constant 102
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 104
call String.appendChar 2
push constant 105
```

```
call String.appendChar 2
push constant 110
call String.appendChar 2
push constant 103
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 102
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 111
call String.appendChar 2
push constant 109
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 117
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 46
call String.appendChar 2
call Output.printString 1
pop temp 0
push constant 1000
call Sys.wait 1
pop temp 0
call Screen.clearScreen 0
pop temp 0
push constant 0
push constant 0
call Output.moveCursor 2
pop temp 0
push constant 0
not
pop local 3
```

```
label IF_FALSE0
goto WHILE_EXP0
label WHILE_END0
push constant 0
pop local 3
push constant 45
call String.new 1
push constant 80
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 79
call String.appendChar 2
push constant 71
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 65
call String.appendChar 2
push constant 77
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 70
call String.appendChar 2
push constant 79
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 69
call String.appendChar 2
push constant 65
call String.appendChar 2
push constant 68
call String.appendChar 2
push constant 73
call String.appendChar 2
```

```
push constant 78
call String.appendChar 2
push constant 71
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 83
call String.appendChar 2
push constant 84
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 73
call String.appendChar 2
push constant 78
call String.appendChar 2
push constant 71
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 70
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 79
call String.appendChar 2
push constant 77
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 84
call String.appendChar 2
push constant 72
call String.appendChar 2
push constant 69
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 75
call String.appendChar 2
push constant 69
```

```
call String.appendChar 2
push constant 89
call String.appendChar 2
push constant 66
call String.appendChar 2
push constant 79
call String.appendChar 2
push constant 65
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 68
call String.appendChar 2
push constant 46
call String.appendChar 2
call Output.printString 1
pop temp 0
call Output.println 0
pop temp 0
push constant 44
call String.new 1
push constant 80
call String.appendChar 2
push constant 108
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 97
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 112
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 101
call String.appendChar 2
```

```
push constant 115
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 39
call String.appendChar 2
push constant 69
call String.appendChar 2
push constant 83
call String.appendChar 2
push constant 67
call String.appendChar 2
push constant 39
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 111
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 109
call String.appendChar 2
push constant 111
call String.appendChar 2
push constant 118
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 111
call String.appendChar 2
push constant 32
```

```
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 111
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 104
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 110
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 120
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 112
call String.appendChar 2
push constant 46
call String.appendChar 2
call Output.printString 1
pop temp 0
call Output.println 0
pop temp 0
call Output.println 0
```

```
pop temp 0
push constant 0
pop local 1
label WHILE_EXP1
push local 3
not
not
if-goto WHILE_END1
push constant 25
call String.new 1
push constant 80
call String.appendChar 2
push constant 108
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 97
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 110
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 104
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 105
call String.appendChar 2
push constant 110
call String.appendChar 2
push constant 103
call String.appendChar 2
push constant 58
call String.appendChar 2
push constant 32
call String.appendChar 2
call Keyboard.readLine 1
pop local 5
call Output.println 0
pop temp 0
push local 5
call Output.printString 1
pop temp 0
call Output.println 0
pop temp 0
call Keyboard.readChar 0
pop local 1
call Output.println 0
pop temp 0
push local 1
push constant 140
eq
if-goto IF_TRUE1
goto IF_FALSE1
label IF_TRUE1
push constant 100
call Sys.wait 1
pop temp 0
call Output.println 0
pop temp 0
call Output.println 0
pop temp 0
push constant 33
call String.new 1
push constant 77
call String.appendChar 2
push constant 111
call String.appendChar 2
push constant 118
call String.appendChar 2
push constant 105
call String.appendChar 2
push constant 110
call String.appendChar 2
push constant 103
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 111
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 97
call String.appendChar 2
push constant 100
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 105
call String.appendChar 2
push constant 110
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 103
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 102
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 111
call String.appendChar 2
push constant 109
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 117
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 46
call String.appendChar 2
call Output.printString 1
pop temp 0
push constant 1000
call Sys.wait 1
pop temp 0
call Screen.clearScreen 0
pop temp 0
call Screen.clearScreen 0
pop temp 0
push constant 0
push constant 0
call Output.moveCursor 2
pop temp 0
push constant 0
not
pop local 3
label IF_FALSE1
goto WHILE_EXP1
label WHILE_END1
push constant 0
pop local 3
push constant 46
call String.new 1
push constant 80
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 79
call String.appendChar 2
push constant 71
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 65
call String.appendChar 2
push constant 77
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 70
call String.appendChar 2
push constant 79
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 69
call String.appendChar 2
push constant 65
call String.appendChar 2
push constant 68
call String.appendChar 2
push constant 73
call String.appendChar 2
push constant 78
call String.appendChar 2
push constant 71
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 73
call String.appendChar 2
push constant 78
call String.appendChar 2
push constant 84
call String.appendChar 2
push constant 69
call String.appendChar 2
push constant 71
call String.appendChar 2
push constant 69
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 70
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 79
call String.appendChar 2
push constant 77
call String.appendChar 2
push constant 32
call String.appendChar 2
```

```
call String.appendChar 2
push constant 79
call String.appendChar 2
push constant 77
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 84
call String.appendChar 2
push constant 72
call String.appendChar 2
push constant 69
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 75
call String.appendChar 2
push constant 69
call String.appendChar 2
push constant 89
call String.appendChar 2
push constant 66
call String.appendChar 2
push constant 79
call String.appendChar 2
push constant 65
call String.appendChar 2
push constant 82
call String.appendChar 2
push constant 68
call String.appendChar 2
push constant 46
call String.appendChar 2
call Output.printString 1
pop temp 0
call Output.println 0
pop temp 0
call Output.println 0
pop temp 0
label WHILE_EXP2
push local 3
```

```
not
not
if-goto WHILE_END2
push constant 26
call String.new 1
push constant 80
call String.appendChar 2
push constant 108
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 97
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 110
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 104
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 105
call String.appendChar 2
```

```
push constant 110
call String.appendChar 2
push constant 116
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 103
call String.appendChar 2
push constant 101
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 58
call String.appendChar 2
push constant 32
call String.appendChar 2
call Keyboard.readInt 1
pop local 6
call Output.println 0
pop temp 0
push local 6
call Output.printInt 1
pop temp 0
call Keyboard.readChar 0
pop local 1
call Output.println 0
pop temp 0
push local 1
push constant 140
eq
if-goto IF_TRUE2
goto IF_FALSE2
label IF_TRUE2
push constant 1000
call Sys.wait 1
pop temp 0
call Output.println 0
pop temp 0
call Output.println 0
pop temp 0
push constant 33
```

```
call String.new 1
push constant 84
call String.appendChar 2
push constant 104
call String.appendChar 2
push constant 97
call String.appendChar 2
push constant 110
call String.appendChar 2
push constant 107
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 121
call String.appendChar 2
push constant 111
call String.appendChar 2
push constant 117
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 102
call String.appendChar 2
push constant 111
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 117
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 105
call String.appendChar 2
push constant 110
call String.appendChar 2
push constant 103
call String.appendChar 2
push constant 32
call String.appendChar 2
```

```
push constant 116
call String.appendChar 2
push constant 104
call String.appendChar 2
push constant 105
call String.appendChar 2
push constant 115
call String.appendChar 2
push constant 32
call String.appendChar 2
push constant 112
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 111
call String.appendChar 2
push constant 103
call String.appendChar 2
push constant 114
call String.appendChar 2
push constant 97
call String.appendChar 2
push constant 109
call String.appendChar 2
push constant 46
call String.appendChar 2
call Output.printString 1
pop temp 0
push constant 0
not
pop local 3
label IF_FALSE2
goto WHILE_EXP2
label WHILE_END2
push constant 0
return
```

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.
- To work with your own keyboard library, one can select and load the complete folder on VM Emulator.

Keyboard	27-06-2022 08:38	JACK File	1 KB
Keyboard	27-06-2022 12:44	VM File	1 KB
Main	27-06-2022 08:37	JACK File	2 KB
Main	27-06-2022 12:44	VM File	11 KB

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 echoes the key to the screen and returns the value of the pressed key.

Virtual Machine Emulator (2.5) - C:\Users\HP\Desktop\Amrita\AMRITA APPS\nand2tetris\nand2tetris\tools\TERM_PROJECT_SEM_2

File View Run Help

Slow Fast Animate: No animation View: Screen Format: Decimal

Program

1217	call	String.appendChar 2
1218	push	constant 109
1219	call	String.appendChar 2
1220	push	constant 46
1221	call	String.appendChar 2
1222	call	Output.printString 1
1223	pop	temp 0
1224	push	constant 0
1225	not	
1226	pop	local 3
	label	Main.main$IF_FALSE2
1227	goto	Main.main$WHILE_EXP2
	label	Main.main$WHILE_END2
1228	push	constant 0
1229	return	

Stack

Call Stack

Static

0	0
1	0
2	0
3	0
4	0

Local

Argument

This

That

Temp

0	0
1	0

```
PROGRAM FOR READING CHARACTERS FROM THE KEYBOARD.
Please press 'ESC' to move to the next step.

Keep pressing any key to start.

The ASCII of currently pressed key is: 65


New character pressed is: A
The ASCII of this character is: 65


New character pressed is: M
The ASCII of this character is: 77
```

Global Stack

256	0
257	0
258	0
259	0
260	0
261	0
262	0
263	0
264	0
265	0
266	0
267	0
268	0
269	0
270	0

RAM

SP:	0	256
LCL:	1	0
ARG:	2	0
THIS:	3	0
THAT:	4	0
Temp0:	5	0
Temp1:	6	0
Temp2:	7	0
Temp3:	8	0
Temp4:	9	0
Temp5:	10	0
Temp6:	11	0
Temp7:	12	0
R13:	13	0
R14:	14	0

Running

Virtual Machine Emulator (2.5) - C:\Users\HP\Desktop\Amrita\AMRITA APPS\nand2tetris\nand2tetris\tools\TERM_PROJECT_SEM_2

File View Run Help

Animate: No animation | View: Screen | Format: Decimal

Program

1217	call	String.appendChar 2
1218	push	constant 109
1219	call	String.appendChar 2
1220	push	constant 46
1221	call	String.appendChar 2
1222	call	Output.printString 1
1223	pop	temp 0
1224	push	constant 0
1225	not	
1226	pop	local 3
	label	Main.main$IF_FALSE2
1227	goto	Main.main$WHILE_EXP2
	label	Main.main$WHILE_END2
1228	push	constant 0
1229	return	

Static

0	0
1	0
2	0
3	0
4	0

Local

Argument

This

That

Temp

0	0
1	0

```
PROGRAM FOR READING CHARACTERS FROM THE KEYBOARD.
Please press 'ESC' to move to the next step.

Keep pressing any key to start.

The ASCII of currently pressed key is: 65


New character pressed is: A
The ASCII of this character is: 65


New character pressed is: M
The ASCII of this character is: 77


New character pressed is: Ä
The ASCII of this character is: 140

Moving to read string from user.
```

Stack

Call Stack

Global Stack

256	0
257	0
258	0
259	0
260	0
261	0
262	0
263	0
264	0
265	0
266	0
267	0
268	0
269	0
270	0

RAM

SP:	0	256
LCL:	1	0
ARG:	2	0
THIS:	3	0
THAT:	4	0
Temp0:	5	0
Temp1:	6	0
Temp2:	7	0
Temp3:	8	0
Temp4:	9	0
Temp5:	10	0
Temp6:	11	0
Temp7:	12	0
R13:	13	0
R14:	14	0

VM IMPLEMENTATION

- 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.

Virtual Machine Emulator (2.5) - C:\Users\HP\Desktop\Amrita\AMRITA APPS\nand2tetris\nand2tetris\tools\TERM_PROJECT_SEM_2

File View Run Help

Slow Fast Animate: No animation View: Screen Format: Decimal

Program

1217	call	String.appendChar 2
1218	push	constant 109
1219	call	String.appendChar 2
1220	push	constant 46
1221	call	String.appendChar 2
1222	call	Output.printString 1
1223	pop	temp 0
1224	push	constant 0
1225	not	
1226	pop	local 3
	label	Main.main$IF_FALSE2
1227	goto	Main.main$WHILE_EXP2
	label	Main.main$WHILE_END2
1228	push	constant 0
1229	return	

Static

0	0
1	0
2	0
3	0
4	0

Local

Argument

This

That

Temp

0	0
1	0

```
PROGRAM FOR READING STRING FROM THE KEYBOARD.
Please press 'ESC' to move to the next step.

Please enter the string: VIKHYAT

VIKHYAT


Moving to read integer from user.
```

Stack

Call Stack

Global Stack

256	0
257	0
258	0
259	0
260	0
261	0
262	0
263	0
264	0
265	0
266	0
267	0
268	0
269	0
270	0

RAM

SP:	0	256
LCL:	1	0
ARG:	2	0
THIS:	3	0
THAT:	4	0
Temp0:	5	0
Temp1:	6	0
Temp2:	7	0
Temp3:	8	0
Temp4:	9	0
Temp5:	10	0
Temp6:	11	0
Temp7:	12	0
R13:	13	0
R14:	14	0

VM IMPLEMENTATION

- 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.

Virtual Machine Emulator (2.5) - C:\Users\HP\Desktop\Amrita\AMRITA APPS\nand2tetris\nand2tetris\tools\TERM_PROJECT_SEM_2

File View Run Help

Slow Fast Animate: No animation View: Screen Format: Decimal

Program

1217	call	String.appendChar 2
1218	push	constant 109
1219	call	String.appendChar 2
1220	push	constant 46
1221	call	String.appendChar 2
1222	call	Output.printString 1
1223	pop	temp 0
1224	push	constant 0
1225	not	
1226	pop	local 3
	label	Main.main$IF_FALSE2
1227	goto	Main.main$WHILE_EXP2
	label	Main.main$WHILE_END2
1228	push	constant 0
1229	return	

Stack

Call Stack

Static

0	0
1	0
2	0
3	0
4	0

Local

Argument

This

That

Temp

0	0
1	0

```
PROGRAM FOR READING INTEGER FROM THE KEYBOARD.

Please enter the integer: 123
123
Please enter the integer: -123
-123
Please enter the integer: F123
0
Please enter the integer: 123F
123

Thank you for using this program.
```

Global Stack

256	0
257	0
258	0
259	0
260	0
261	0
262	0
263	0
264	0
265	0
266	0
267	0
268	0
269	0
270	0

RAM

SP:	0	256
LCL:	1	0
ARG:	2	0
THIS:	3	0
THAT:	4	0
Temp0:	5	0
Temp1:	6	0
Temp2:	7	0
Temp3:	8	0
Temp4:	9	0
Temp5:	10	0
Temp6:	11	0
Temp7:	12	0
R13:	13	0
R14:	14	0

PART-2

Topic

- 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

Assembler.py > Assembler > get_hack_file

```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

Assembler.py > Assembler > get_hack_file

```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))

```

```python

    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))


if __name__ == '__main__':
    if len(sys.argv) != 2:
        print("Usage: python Assembler.py Program.asm")
    else:
        asm_file = sys.argv[1]

    hack_assembler = Assembler()
    hack_assembler.assemble(asm_file)
    print('hack file for selected .asm file created')


```

Creating .hack for our keyboard library and Main file

File Edit Selection View Go Run Terminal Help

Assembler.py - Assembler - Visual Studio Code

EXPLORER

OPEN EDITORS

Assembler.py

ASSEMBLER

__pycache__

Assembler.py

Code.py

Keyboard.asm

Keyboard.hack

Lex.py

Main.asm

Main.hack

Parser.py

Part 2 & 3.zip

SymbolTable.py

Assembler.py > ...

```python
        """
        self.pass_1(file)
        self.pass_2(file, self.get_hack_file(file))


if __name__ == '__main__':
    if len(sys.argv) != 2:
        print("Usage: python Assembler.py Program.asm")
    else:
        asm_file = sys.argv[1]

    hack_assembler = Assembler()
    hack_assembler.assemble(asm_file)
    print('hack file for selected .asm file created')
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL JUPYTER

```
PS D:\CODING\Python\EOC_Assignment\Assembler> python .\Assembler.py .\Keyboard.asm
hack file for selected .asm file created
PS D:\CODING\Python\EOC_Assignment\Assembler> python .\Assembler.py .\Main.asm
hack file for selected .asm file created
PS D:\CODING\Python\EOC_Assignment\Assembler>
```

Comparing our KEYBOARD assembler file with the IN-BUILT assembler

Assembler (2.5) - D:\CODING\Python\EOC_Assignment\Assembler\Keyboard.asm

File Run Help

Source

```
//function Keyboard.new 0
(Keyboard.new)
//push constant 1
@1
D=A
@SP
A=M
M=D
@SP
M=M+1
//call Memory.alloc 1
@Memory.alloc$ret.0
D=A
@SP
A=M
M=D
@SP
M=M+1
@LCL
D=M
@SP
A=M
M=D
@SP
M=M+1
@ARG
D=M
@SP
A=M
M=D
```

Destination

```
0000000000000001
1110110000010000
0000000000000000
1111110000100000
1110001100001000
0000000000000000
1111110111001000
0000000000110101
1110110000010000
0000000000000000
1111110000100000
1110001100001000
0000000000000000
1111110111001000
0000000000000001
1111110000010000
0000000000000000
1111110000100000
1110001100001000
0000000000000000
1111110111001000
0000000000000010
1111110000010000
0000000000000000
1111110000100000
1110001100001000
0000000000000000
1111110111001000
0000000000000011
1111110000010000
```

Comparison

```
0000000000000001
1110110000010000
0000000000000000
1111110000100000
1110001100001000
0000000000000000
1111110111001000
0000000000110101
1110110000010000
0000000000000000
1111110000100000
1110001100001000
0000000000000000
1111110111001000
0000000000000001
1111110000010000
0000000000000000
1111110000100000
1110001100001000
0000000000000000
1111110111001000
0000000000000010
1111110000010000
0000000000000000
1111110000100000
1110001100001000
0000000000000000
1111110111001000
0000000000000011
1111110000010000
```

File compilation & comparison succeeded

Comparing our MAIN assembler file with the IN-BUILT assembler

Assembler (2.5) - D:\CODING\Python\EOC_Assignment\Assembler\Main.asm

File Run Help

Source

```
//function Keyboard.new 0
(Keyboard.new)
//push constant 1
@1
D=A
@SP
A=M
M=D
@SP
M=M+1
//call Memory.alloc 1
@Memory.alloc$ret.0
D=A
@SP
A=M
M=D
@SP
M=M+1
@LCL
D=M
@SP
A=M
M=D
@SP
M=M+1
@ARG
D=M
@SP
A=M
M=D
```

Destination

```
0000000000000001
1110110000010000
0000000000000000
1111110000100000
1110001100001000
0000000000000000
1111110111001000
0000000000110101
1110110000010000
0000000000000000
1111110000100000
1110001100001000
0000000000000000
1111110111001000
0000000000000001
1111110000010000
0000000000000000
1111110000100000
1110001100001000
0000000000000000
1111110111001000
0000000000000010
1111110000010000
0000000000000000
1111110000100000
1110001100001000
0000000000000000
1111110111001000
0000000000000011
1111110000010000
```

Comparison

```
0000000000000001
1110110000010000
0000000000000000
1111110000100000
1110001100001000
0000000000000000
1111110111001000
0000000000110101
1110110000010000
0000000000000000
1111110000100000
1110001100001000
0000000000000000
1111110111001000
0000000000000001
1111110000010000
0000000000000000
1111110000100000
1110001100001000
0000000000000000
1111110111001000
0000000000000010
1111110000010000
0000000000000000
1111110000100000
1110001100001000
0000000000000000
1111110111001000
0000000000000011
1111110000010000
```

File compilation & comparison succeeded

Backup FILE!!

- Note: Firstly, one must convert .VM file to .asm file using a VM translator and then start working with assembler. We have to keep all the files in the same folder.

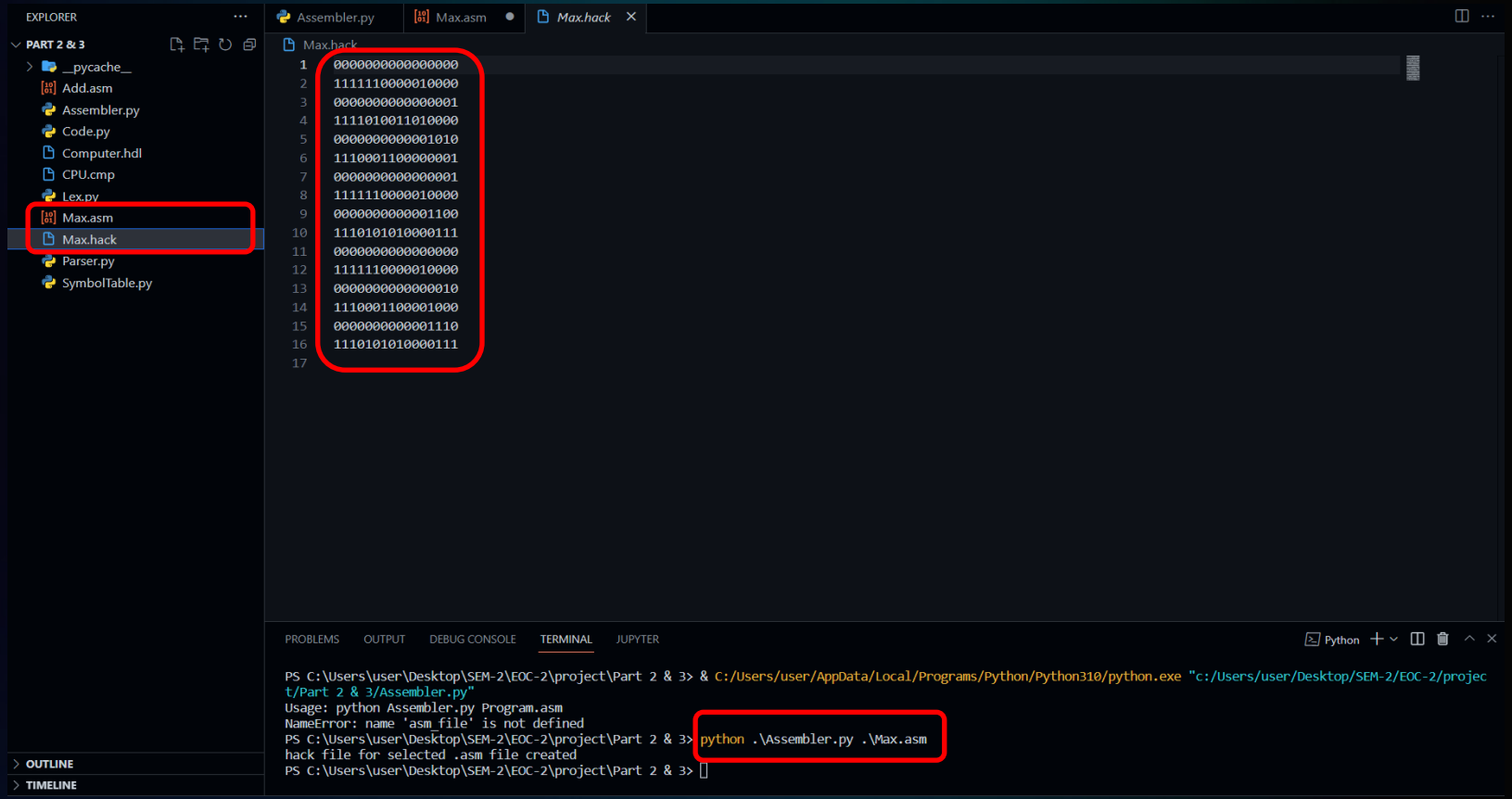
PART 2 & 3

- __pycache__
- Add.asm
- Assembler.py
- Code.py
- Computer.hdl
- CPU.cmp
- Lex.py
- Max.asm
- Parser.py
- SymbolTable.py

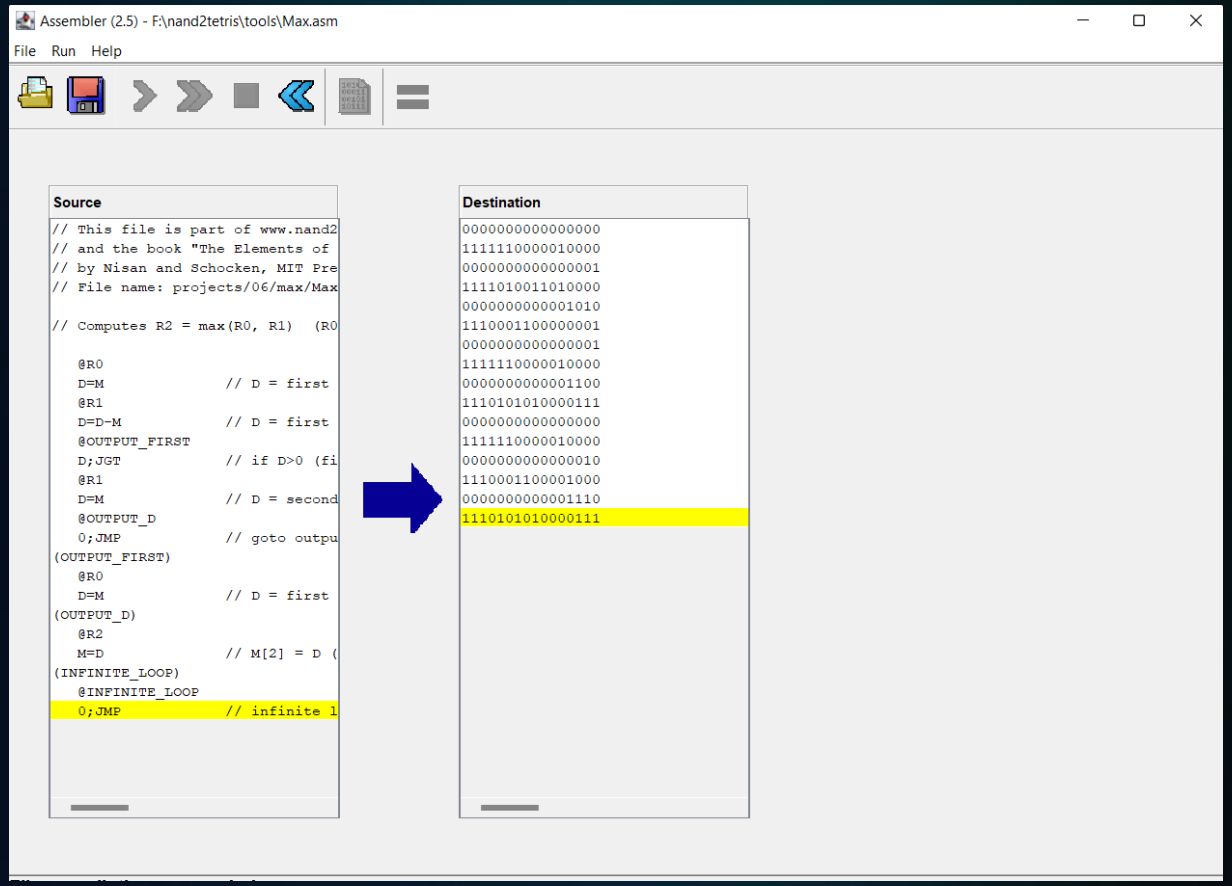
Assembler.py > ...

```python
    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'

    def _get_address(self, symbol):
        """
        Helper method. Looks-up the address of a symbol (decimal value, label or variable).
```

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



- Converting .asm file to .hack file using nand2tetris software, by using assembler tool.



- Both the hack files that we obtained using our own assembler and nand2teris are similar.

```
0000000000000000
1111110000010000
0000000000000001
1111010011010000
0000000000001010
1110001100000001
0000000000000001
1111110000010000
0000000000001100
1110101010000111
0000000000000000
1111110000010000
0000000000000010
1110001100001000
0000000000001110
1110101010000111
```

Destination

```
0000000000000000
1111110000010000
0000000000000001
1111010011010000
0000000000001010
1110001100000001
0000000000000001
1111110000010000
0000000000001100
1110101010000111
0000000000000000
1111110000010000
0000000000000010
1110001100001000
0000000000001110
1110101010000111
```

PART-3

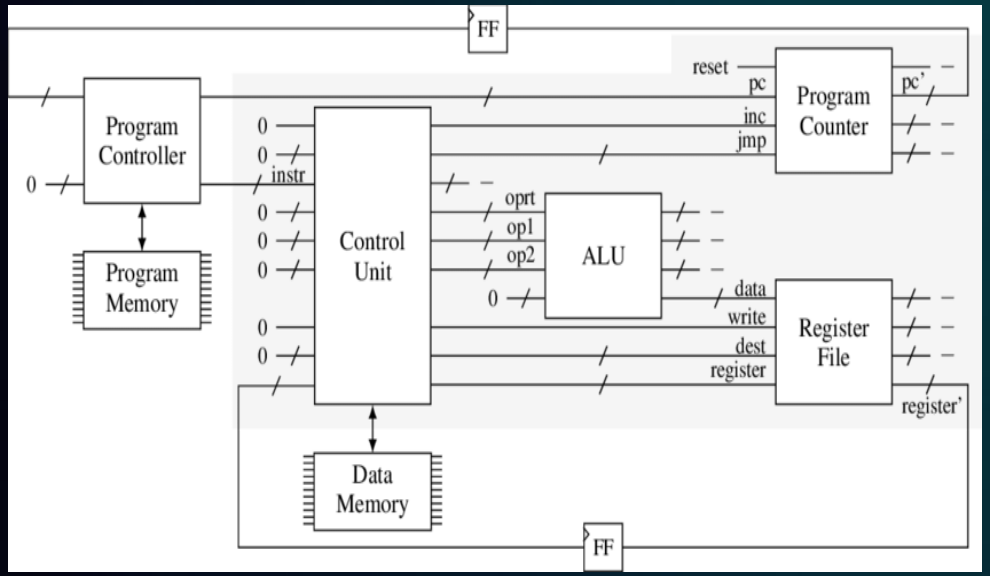
Topic

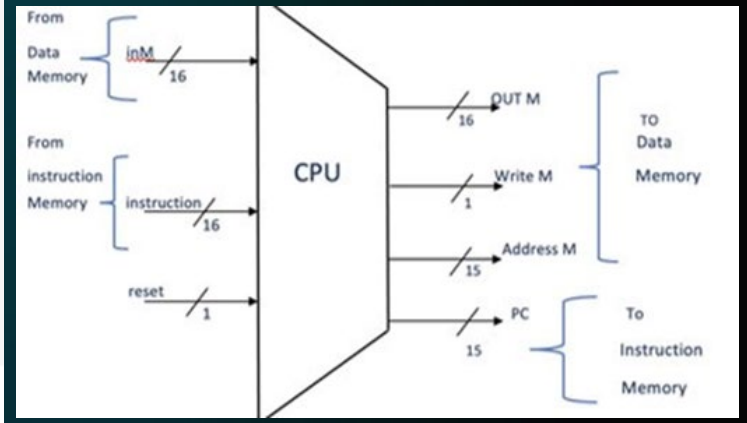
- Dump the machine codes generated by your own assembler into the Computer.hdl. Then design the architecture for implementing your machine instructions to get the result. (Here I presume that you already have the hdl scripts of all the hardware's used by the CPU. Additional thing what you have to do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

CENTRAL PROCESSING UNIT

- 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.

What does it mean 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);
}
```

Working with .hack file of KEYBOARD & MAIN

Hardware Simulator (2.5) - C:\Users\HP\Desktop\Amrita\AMRITA APPS\nand2tetris\nand2tetris\projects\05\Computer.hdl

File View Run Help

Slow Fast Animate: No animation Format: Decimal View: Screen

Chip Name : Computer (Clocked) Time : 56555+

Input pins

Name	Value
reset	0

Output pins

Name	Value

HDL

```
/**
 * The HACK computer, including
 * When reset is 0, the program
 * When reset is 1, the executi
 * Thus, to start a program's e
 * and "down" (0). From this pc
 * the software. In particular,
 * screen may show some output
 * with the computer via the ke
 */

CHIP Computer {
    IN reset;
```

Internal pins

Name	Value
pc[15]	0
instruction[16]	1
inM[16]	0
outM[16]	0
writeM	0
addressM[15]	0

RAM 16K:

0	6112
1	6111
2	6105
3	0
4	53
5	0
6	53

ROM: Bin

40	0000000000000000
41	1111110111001000
42	1111110000010000
43	0000000000000110
44	1110010011010000
45	0000000000000010
46	1110001100001000

A: 6099 D: 6113 PC: 46

ALU

D Input : 6093

M/A Input : 3

M

ALU output : 53

Running...

Working with .hack file of KEYBOARD & MAIN

Hardware Simulator (2.5) - C:\Users\HP\Desktop\Amrita\AMRITA APPS\nand2tetris\nand2tetris\projects\05\Computer.hdl

File View Run Help

Slow Fast Animate: No animation Format: Decimal View: Screen

Chip Name : Computer (Clocked) Time : 191962+

Input pins

Name	Value
reset	0

Output pins

Name	Value

HDL

```
/**
 * The HACK computer, including
 * When reset is 0, the program
 * When reset is 1, the executi
 * Thus, to start a program's e
 * and "down" (0). From this pc
 * the software. In particular,
 * screen may show some output
 * with the computer via the ke
 */

CHIP Computer {
    IN reset;
```

Internal pins

Name	Value
pc[15]	0
instruction[16]	1
inM[16]	0
outM[16]	0
writeM	0
addressM[15]	0

RAM 16K:

13	0
14	53
15	15
16	9
17	0
18	53
19	19

ROM: Bin

16	0000000000000000
17	1111110000100000
18	1110001100001000
19	0000000000000000
20	1111110111001000
21	0000000000000010
22	1111110000010000

A: 0 D: 20735 PC: 23

ALU

D Input : 53

M/A Input : 20753

D&M

ALU output : 16

Running...

Working with .hack file of KEYBOARD & MAIN

Hardware Simulator (2.5) - C:\Users\HP\Desktop\Amrita\AMRITA APPS\nand2tetris\nand2tetris\projects\05\Computer.hdl

File View Run Help

Slow Fast Animate: No animation Format: Decimal View: Screen

Chip Name : Computer (Clocked) Time : 264378

Input pins

Name	Value
reset	0

Output pins

Name	Value

HDL

```
/**
 * The HACK computer, including
 * When reset is 0, the program
 * When reset is 1, the executi
 * Thus, to start a program's e
 * and "down" (0). From this pc
 * the software. In particular,
 * screen may show some output
 * with the computer via the ke
 */

CHIP Computer {
    IN reset;
```

Internal pins

Name	Value
pc[15]	0
instruction[16]	1
inM[16]	0
outM[16]	0
writeM	0
addressM[15]	0

RAM 16K:

0	28582
1	28579
2	28573
3	0
4	53
5	0
6	53

ROM: Bin

44	1110010011010000
45	0000000000000010
46	1110001100001000
47	0000000000000000
48	1111110000010000
49	0000000000000001
50	1110001100001000

A: 6 D: 28581 PC: 23

ALU

D Input : 0

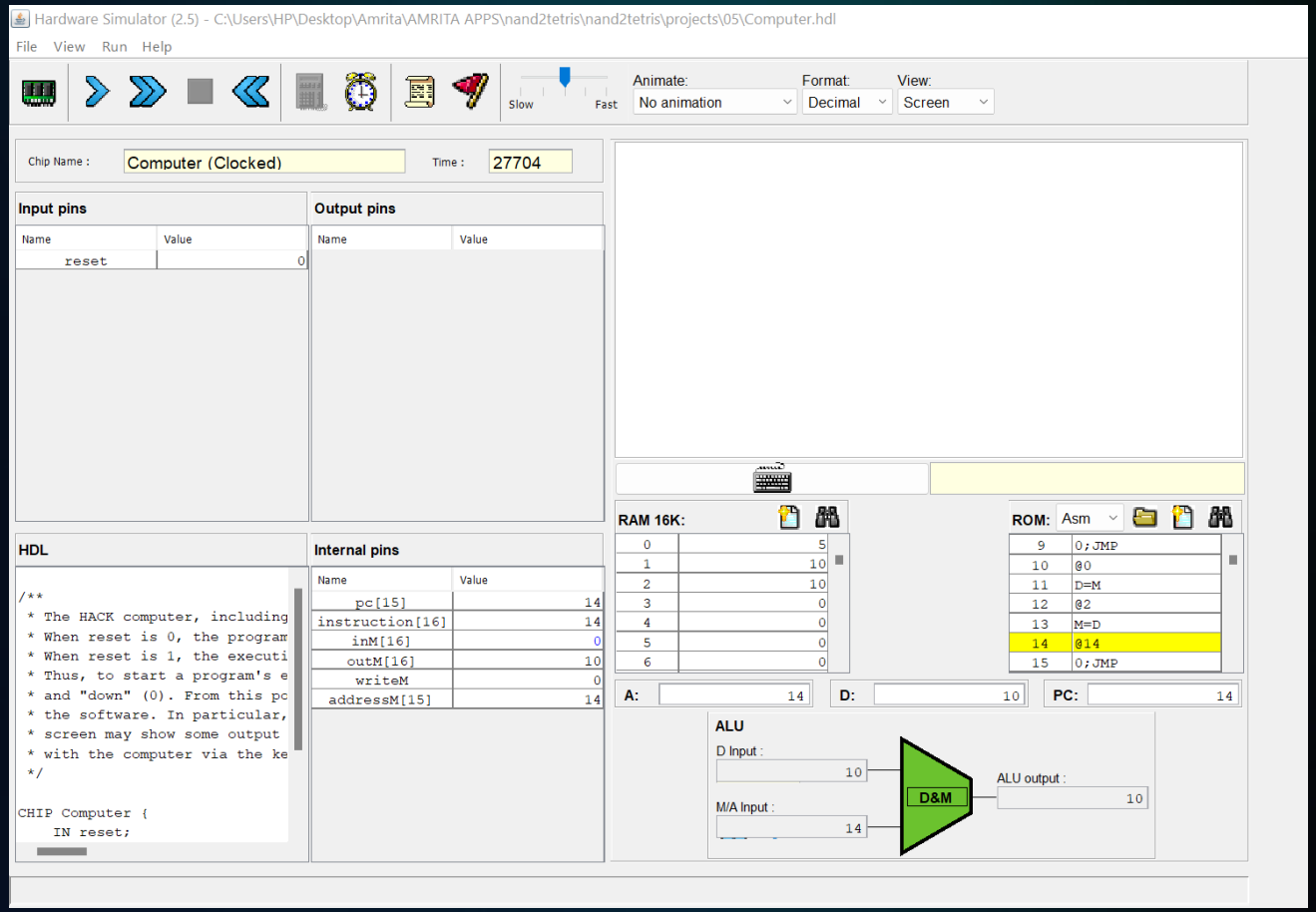
M/A Input : 28587

M

ALU output : 28575

Running...

ALTERNATE CODE[MAX ASM]



ALTERNATE CODE[MAX HACK]

Hardware Simulator (2.5) - C:\Users\HP\Desktop\Amrita\AMRITA APPS\nand2tetris\nand2tetris\projects\05\Computer.hdl

File View Run Help

Slow Fast

Animate: No animation

Format: Decimal

View: Screen

Chip Name : Computer (Clocked)

Time : 10754+

Input pins

Name	Value
reset	0

Output pins

Name	Value

HDL

```
/**
 * The HACK computer, including
 * When reset is 0, the program
 * When reset is 1, the executi
 * Thus, to start a program's e
 * and "down" (0). From this pc
 * the software. In particular,
 * screen may show some output
 * with the computer via the ke
 */

CHIP Computer {
    IN reset;
```

Internal pins

Name	Value
pc[15]	14
instruction[16]	14
inM[16]	0
outM[16]	2
writeM	0
addressM[15]	14

RAM 16K:

0	9
1	18
2	18
3	0
4	0
5	0
6	0

ROM: Bin

9	1110101010000111
10	0000000000000000
11	1111110000010000
12	0000000000000010
13	1110001100001000
14	0000000000001110
15	1110101010000111

A: 14 D: 18 PC: 15

ALU

D Input : 18

M/A Input : 14

D&M

ALU output : 2

CONCLUSION

In this project we have implemented and shown

1. How a Keyboard works
2. How a Hack Computer works
3. How high level language is abstracted at machine level.

THANK YOU