**22AIE113- Elements of Computing – 2**

**Assembler:**

**import** **java.io.BufferedReader**;

**import** **java.io.FileReader**;

**import** **java.io.FileWriter**;

**import** **java.io.IOException**;

**import** **java.io.FileWriter**;

**import** **java.util.\***;

**public** **class** assembler {

**static** **String** **C2Binary**(**String** ***instruction***){

**HashMap**<**String**, **String**> destDict **=** **new** **HashMap**<>();

        destDict.**put**("", "000");

        destDict.**put**("M", "001");

        destDict.**put**("D", "010");

        destDict.**put**("MD", "011");

        destDict.**put**("A", "100");

        destDict.**put**("AM", "101");

        destDict.**put**("AD", "110");

        destDict.**put**("AMD", "111");

**HashMap**<**String**, **String**> compDict **=** **new** **HashMap**<>();

        compDict.**put**("0", "0101010");

        compDict.**put**("1", "0111111");

        compDict.**put**("-1", "0111010");

        compDict.**put**("D", "0001100");

        compDict.**put**("A", "0110000");

        compDict.**put**("!D", "0001101");

        compDict.**put**("!A", "0110001");

        compDict.**put**("-D", "0001111");

        compDict.**put**("-A", "0110011");

        compDict.**put**("D+1", "0011111");

        compDict.**put**("A+1", "0110111");

        compDict.**put**("D-1", "0001110");

        compDict.**put**("A-1", "0110010");

        compDict.**put**("D+A", "0000010");

        compDict.**put**("A+D", "0000010");

        compDict.**put**("D-A", "0010011");

        compDict.**put**("A-D", "0000111");

        compDict.**put**("D&A", "0000000");

        compDict.**put**("A&D", "0000000");

        compDict.**put**("D|A", "0010101");

        compDict.**put**("A|D", "0010101");

        compDict.**put**("M", "1110000");

        compDict.**put**("!M", "1110001");

        compDict.**put**("-M", "1110011");

        compDict.**put**("M+1", "1110111");

        compDict.**put**("M-1", "1110010");

        compDict.**put**("D+M", "1000010");

        compDict.**put**("D-M", "1010011");

        compDict.**put**("M-D", "1000111");

        compDict.**put**("D&M", "1000000");

        compDict.**put**("C|M", "1010101");

**HashMap**<**String**, **String**> jumpDict **=** **new** **HashMap**<>();

        jumpDict.**put**("", "000");

        jumpDict.**put**("JGT", "001");

        jumpDict.**put**("JEQ", "010");

        jumpDict.**put**("JGE", "011");

        jumpDict.**put**("JLT", "100");

        jumpDict.**put**("JNE", "101");

        jumpDict.**put**("JLE", "110");

        jumpDict.**put**("JMP", "111");

**String**[] destAndCompJump **=** instruction.**split**("=");

**String** dest **=** destAndCompJump.length **>** 1 **?** destAndCompJump[0] **:** "";

**String** compJump **=** destAndCompJump[destAndCompJump.length **-** 1];

**String**[] compAndJump **=** compJump.**split**(";");

**String** comp **=** compAndJump[0];

**String** jump **=** compAndJump.length **>** 1 **?** compAndJump[1] **:** "";

**String** binary **=** "111" **+** compDict.**get**(comp) **+**  destDict.**get**(dest) **+**jumpDict.**get**(jump);

**return** binary;

    }

**public** **static** **void** **main**(**String**[] ***args***) **throws** **IOException** {

**BufferedReader** reader **=** **new** **BufferedReader**(**new** **FileReader**("Rect.asm"));

**List**<**String**> lines **=** **new** **ArrayList**<>();

**String** line **=** reader.**readLine**();

**while** (line **!=** null) {

            lines.**add**(line);

            line **=** reader.**readLine**();

        }

        reader.**close**();

**ArrayList**<**String**> preDefinedSymbols **=** **new** **ArrayList**();

        preDefinedSymbols.**add**("R0");

        preDefinedSymbols.**add**("R1");

        preDefinedSymbols.**add**("R2");

        preDefinedSymbols.**add**("R3");

        preDefinedSymbols.**add**("R4");

        preDefinedSymbols.**add**("R5");

        preDefinedSymbols.**add**("R6");

        preDefinedSymbols.**add**("R7");

        preDefinedSymbols.**add**("R8");

        preDefinedSymbols.**add**("R9");

        preDefinedSymbols.**add**("R10");

        preDefinedSymbols.**add**("R11");

        preDefinedSymbols.**add**("R12");

        preDefinedSymbols.**add**("R13");

        preDefinedSymbols.**add**("R14");

        preDefinedSymbols.**add**("R15");

**HashMap**<**String**, **Integer**> unqLbls **=** **new** **HashMap**<>();

        unqLbls.**put**("SCREEN", 16384);

        unqLbls.**put**("KBD", 24576);

        unqLbls.**put**("SP", 0);

        unqLbls.**put**("LCL", 1);

        unqLbls.**put**("ARG", 2);

        unqLbls.**put**("THIS", 3);

        unqLbls.**put**("THAT", 4);

**ArrayList**<**String**> symnVar **=** **new** **ArrayList**<>();

**ArrayList**<**String**> noWhiteSpace **=** **new** **ArrayList**<>();

**ArrayList**<**String**> noLblBrackets **=** **new** **ArrayList**<>();

**ArrayList**<**String**> labels **=** **new** **ArrayList**<>();

**ArrayList**<**String**> aInstructions **=** **new** **ArrayList**<>();

**ArrayList**<**String**> modifiedNoWhiteSpace **=** **new** **ArrayList**<>();

**ArrayList**<**String**> hackFile **=** **new** **ArrayList**<>();

**HashMap**<**String**, **Integer**> symbolTable **=** **new** **HashMap**<>();

**ArrayList**<**String**> numbersForAins **=** **new** **ArrayList**<>();

**for** (**int** i **=** 0; i **<** 32768; i**++**) {

            numbersForAins.**add**(Integer.**toString**(i));

        }

**for** (**String** l **:** lines) {

**String** v;

**if** (l.**equals**("")) **continue**;

**if** (l.**contains**(" ")) l**=**l.**replace**(" ", "");

**if** (l.**equals**("\n") **||** l.**charAt**(0) **==** '/') { **continue**; }

**else** {

                l **=** l.**replaceAll**("[\\n\\t\\s]", "");

**if** (l.**indexOf**("/") **!=** **-**1) { v **=** l.**substring**(0, l.**indexOf**("/"));}

**else** { v **=** l; }

            }

            noWhiteSpace.**add**(v);

        }

*// System.out.println(noWhiteSpace);*

**for** (**String** s **:** noWhiteSpace) {

            s **=** s.**replaceAll**("[\\[\\]\\(\\)\\'\\ ]", "");

            noLblBrackets.**addAll**(Arrays.**asList**(s.**split**(",")));

        }

*// System.out.println(noLblBrackets);*

**int** lbl**=**0;

**for**(**String** s **:** noWhiteSpace){

**if** (s.**charAt**(0)**==**'('){

                s **=** s.**replace**("(", "");

                s **=** s.**replace**(")","");

                labels.**add**(lbl, s);

                lbl **+=** 1;

            }

        }

*// System.out.println(labels);*

**for** (**String** s **:** noWhiteSpace){

**if** (s.**charAt**(0)**==**'@'){

                s **=** s.**replace**("@", "");

**if** (**!**Arrays.**asList**(aInstructions).**contains**(s)) {

                    aInstructions.**add**(s);

**if** (**!**numbersForAins.**contains**(s) **&&** **!**symnVar.**contains**(s)) {

                        symnVar.**add**(s);

                    }

**if** (**!**numbersForAins.**contains**(s) **&&** **!**labels.**contains**(s) **&&** **!**preDefinedSymbols.**contains**(s) **&&** **!**unqLbls.**containsKey**(s)){

                        preDefinedSymbols.**add**(s);

                    }

                }

            }

        }

*// System.out.println(preDefinedSymbols);*

**for** (**String** s **:** symnVar){

**if** (preDefinedSymbols.**contains**(s)){

                symbolTable.**put**(s, preDefinedSymbols.**indexOf**(s));

            }

**if** (labels.**contains**(s)){

                symbolTable.**put**(s, (noLblBrackets.**indexOf**(s)**-**labels.**indexOf**(s)));

            }

**if** (unqLbls.**containsKey**(s)){

**int** k **=** (unqLbls.**get**(s));

                symbolTable.**put**(s,k);

            }

        }

*// System.out.println(unqLbls);*

*// System.out.println(symbolTable);*

**for** (**String** s**:**noWhiteSpace){

**if** (s.**contains**("(")){**continue**;}

**if** (s.**contains**("@")){

                s **=** s.**replace**("@", "");

**if** (**!**numbersForAins.**contains**(s)){s **=** Integer.**toString**(symbolTable.**get**(s));}

**int** S **=** Integer.**parseInt**(s);

**String** Bin\_a **=** Integer.**toBinaryString**(S);

**String** Bin\_A**=**"";

**for** (**int** i**=**0;i**<**16**-**Bin\_a.**length**();i**++**){

                    Bin\_A **=** Bin\_A**+**"0";

                }

                Bin\_A **=** Bin\_A **+** Bin\_a;

                hackFile.**add**(Bin\_A);

                s **=** '@' **+** s;

            }

**else**{

                hackFile.**add**(**C2Binary**(s).**toString**());

            }

            modifiedNoWhiteSpace.**add**(s);

        }

**FileWriter** Hfile **=** **new** **FileWriter**("HackFile.hack");

**for**(**String** s **:**hackFile){

            System.out.**println**(s);

            Hfile.**write**(s);

            Hfile.**append**("\n");

        }

        Hfile.**close**();;

    }

}

**Rect.asm file:**

// This file is part of www.nand2tetris.org

// and the book "The Elements of Computing Systems"

// by Nisan and Schocken, MIT Press.

// File name: projects/06/rect/Rect.asm

// Draws a rectangle at the top-left corner of the screen.

// The rectangle is 16 pixels wide and R0 pixels high.

@0

D=M

@INFINITE\_LOOP

D;JLE

@counter

M=D

@SCREEN

D=A

@address

M=D

(LOOP)

@address

A=M

M=-1

@address

D=M

@32

D=D+A

@address

M=D

@counter

MD=M-1

@LOOP

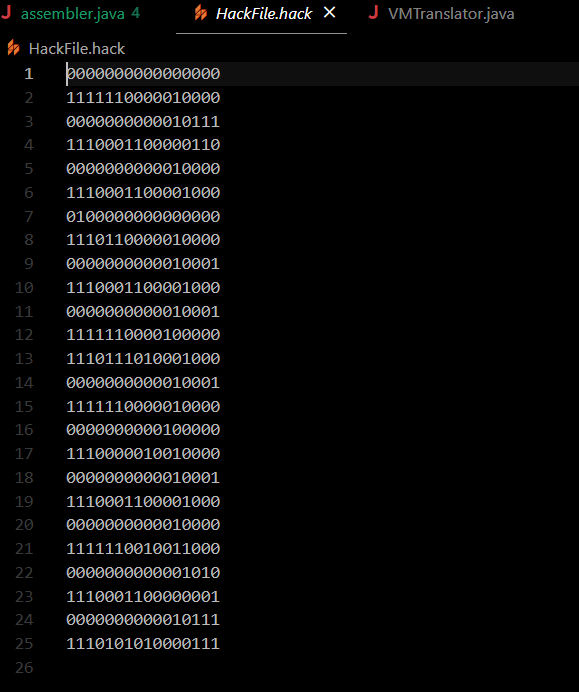
D;JGT

(INFINITE\_LOOP)

@INFINITE\_LOOP

0;JMP

**Output Hack file:**

****

**VM Translator:**

**Code:**

**import** **java.io.\***;

**public** **class** VMTranslator {

**private** **static** **int** labelCounter **=** 0;

**private** **static** **String** **pushConstant**(**String** ***i***) {

**String** res **=** "";

        res **+=** "@" **+** i **+** "\n";

        res **+=** "D=A" **+** "\n";

        res **+=** "@SP" **+** "\n";

        res **+=** "A=M" **+** "\n";

        res **+=** "M=D" **+** "\n";

        res **+=** "@SP" **+** "\n";

        res **+=** "M=M+1" **+** "\n";

**return** res;

    }

**private** **static** **String** **pushStatic**(**String** ***i***) {

**String** res **=** "";

        res **+=** "@" **+** i **+** "\n";

        res **+=** "D=M" **+** "\n";

        res **+=** "@SP" **+** "\n";

        res **+=** "A=M" **+** "\n";

        res **+=** "M=D" **+** "\n";

        res **+=** "@SP" **+** "\n";

        res **+=** "M=M+1" **+** "\n";

**return** res;

    }

**private** **static** **String** **pushPointer**(**String** ***i***) {

**String** res **=** "";

**if** (i.**equals**("0")) {

            res **+=** "@THIS" **+** "\n";

        } **else** **if** (i.**equals**("1")) {

            res **+=** "@THAT" **+** "\n";

        }

        res **+=** "D=M" **+** "\n";

        res **+=** "@SP" **+** "\n";

        res **+=** "A=M" **+** "\n";

        res **+=** "M=D" **+** "\n";

        res **+=** "@SP" **+** "\n";

        res **+=** "M=M+1" **+** "\n";

**return** res;

    }

**private** **static** **String** **pushSegment**(**String** ***segment***, **String** ***i***) {

**String** res **=** "";

        res **+=** "@" **+** i **+** "\n";

        res **+=** "D=A" **+** "\n";

**if** (segment.**equals**("local")) {

            res **+=** "@LCL" **+** "\n";

        } **else** **if** (segment.**equals**("argument")) {

            res **+=** "@ARG" **+** "\n";

        } **else** **if** (segment.**equals**("this")) {

            res **+=** "@THIS" **+** "\n";

        } **else** **if** (segment.**equals**("that")) {

            res **+=** "@THAT" **+** "\n";

        } **else** **if** (segment.**equals**("temp")) {

            res **+=** "@R5" **+** "\n";

        }

        res **+=** "A=D+M" **+** "\n";

        res **+=** "D=M" **+** "\n";

        res **+=** "@SP" **+** "\n";

        res **+=** "A=M" **+** "\n";

        res **+=** "M=D" **+** "\n";

        res **+=** "@SP" **+** "\n";

        res **+=** "M=M+1" **+** "\n";

**return** res;

    }

**private** **static** **String** **popStatic**(**String** ***i***) {

**String** res **=** "";

        res **+=** "@SP" **+** "\n";

        res **+=** "AM=M-1" **+** "\n";

        res **+=** "D=M" **+** "\n";

        res **+=** "@" **+** i **+** "\n";

        res **+=** "M=D" **+** "\n";

**return** res;

    }

**private** **static** **String** **popSegment**(**String** ***segment***, **String** ***i***) {

**String** res **=** "";

        res **+=** "@" **+** i **+** "\n";

        res **+=** "D=A" **+** "\n";

**if** (segment.**equals**("local")) {

            res **+=** "@LCL" **+** "\n";

        } **else** **if** (segment.**equals**("argument")) {

            res **+=** "@ARG" **+** "\n";

        } **else** **if** (segment.**equals**("this")) {

            res **+=** "@THIS" **+** "\n";

        } **else** **if** (segment.**equals**("that")) {

            res **+=** "@THAT" **+** "\n";

        } **else** **if** (segment.**equals**("temp")) {

            res **+=** "@R5" **+** "\n";

        }

        res **+=** "D=D+M" **+** "\n";

        res **+=** "@R13" **+** "\n";

        res **+=** "M=D" **+** "\n";

        res **+=** "@SP" **+** "\n";

        res **+=** "AM=M-1" **+** "\n";

        res **+=** "D=M" **+** "\n";

        res **+=** "@R13" **+** "\n";

        res **+=** "A=M" **+** "\n";

        res **+=** "M=D" **+** "\n";

**return** res;

    }

**private** **static** **String** **popPointer**(**String** ***i***) {

**String** res **=** "";

        res **+=** "@SP" **+** "\n";

        res **+=** "AM=M-1" **+** "\n";

        res **+=** "D=M" **+** "\n";

**if** (i.**equals**("0")) {

            res **+=** "@THIS" **+** "\n";

        } **else** **if** (i.**equals**("1")) {

            res **+=** "@THAT" **+** "\n";

        }

        res **+=** "M=D" **+** "\n";

**return** res;

    }

**private** **static** **String** **arithSegment**(**String** ***segment***) {

**String** res **=** "";

**if** (segment.**equals**("add")) {

            res **+=** "@SP" **+** "\n" **+** "AM=M-1" **+** "\n" **+** "D=M" **+** "\n" **+** "A=A-1" **+** "\n" **+** "M=M+D" **+** "\n";

        } **else** **if** (segment.**equals**("sub")) {

            res **+=** "@SP" **+** "\n" **+** "AM=M-1" **+** "\n" **+** "D=M" **+** "\n" **+** "A=A-1" **+** "\n" **+** "M=M-D" **+** "\n";

        } **else** **if** (segment.**equals**("neg")) {

            res **+=** "@SP" **+** "\n" **+** "A=M-1" **+** "\n" **+** "M=-M" **+** "\n";

        } **else** **if** (segment.**equals**("eq")) {

            res **+=** "@SP" **+** "\n" **+** "AM=M-1" **+** "\n" **+** "D=M" **+** "\n" **+** "A=A-1" **+** "\n"

**+** "D=M-D" **+** "\n" **+** "@EQ\_TRUE" **+** labelCounter **+** "\n" **+** "D;JEQ" **+** "\n" **+** "@SP" **+** "\n" **+** "A=M-1" **+** "\n"

**+** "M=0" **+** "\n" **+** "@EQ\_END" **+** labelCounter **+** "\n" **+** "0;JMP" **+** "\n" **+** "(EQ\_TRUE" **+** labelCounter **+** ")\n"

**+** "@SP" **+** "\n" **+** "A=M-1" **+** "\n" **+** "M=-1" **+** "\n" **+** "(EQ\_END" **+** labelCounter **+** ")\n";

            labelCounter**++**;

        } **else** **if** (segment.**equals**("gt")) {

            res **+=** "@SP" **+** "\n" **+** "AM=M-1" **+** "\n" **+** "D=M" **+** "\n" **+** "A=A-1" **+** "\n"

**+** "D=M-D" **+** "\n" **+** "@GT\_TRUE" **+** labelCounter **+** "\n" **+** "D;JGT" **+** "\n" **+** "@SP" **+** "\n" **+** "A=M-1" **+** "\n"

**+** "M=0" **+** "\n" **+** "@GT\_END" **+** labelCounter **+** "\n" **+** "0;JMP" **+** "\n" **+** "(GT\_TRUE" **+** labelCounter **+** ")\n"

**+** "@SP" **+** "\n" **+** "A=M-1" **+** "\n" **+** "M=-1" **+** "\n" **+** "(GT\_END" **+** labelCounter **+** ")\n";

            labelCounter**++**;

        } **else** **if** (segment.**equals**("lt")) {

            res **+=** "@SP" **+** "\n" **+** "AM=M-1" **+** "\n" **+** "D=M" **+** "\n" **+** "A=A-1" **+** "\n"

**+** "D=M-D" **+** "\n" **+** "@LT\_TRUE" **+** labelCounter **+** "\n" **+** "D;JLT" **+** "\n" **+** "@SP" **+** "\n" **+** "A=M-1" **+** "\n"

**+** "M=0" **+** "\n" **+** "@LT\_END" **+** labelCounter **+** "\n" **+** "0;JMP" **+** "\n" **+** "(LT\_TRUE" **+** labelCounter **+** ")\n"

**+** "@SP" **+** "\n" **+** "A=M-1" **+** "\n" **+** "M=-1" **+** "\n" **+** "(LT\_END" **+** labelCounter **+** ")\n";

            labelCounter**++**;

        } **else** **if** (segment.**equals**("and")) {

            res **+=** "@SP" **+** "\n" **+** "AM=M-1" **+** "\n" **+** "D=M" **+** "\n" **+** "A=A-1" **+** "\n" **+** "M=D&M" **+** "\n";

        } **else** **if** (segment.**equals**("or")) {

            res **+=** "@SP" **+** "\n" **+** "AM=M-1" **+** "\n" **+** "D=M" **+** "\n" **+** "A=A-1" **+** "\n" **+** "M=D|M" **+** "\n";

        } **else** **if** (segment.**equals**("not")) {

            res **+=** "@SP" **+** "\n" **+** "A=M-1" **+** "\n" **+** "M=!M" **+** "\n";

        }

**return** res;

    }

**private** **static** **String** **label**(**String** ***label***) {

**return** "(" **+** label **+** ")\n";

    }

**private** **static** **String** **gotoLabel**(**String** ***label***) {

**return** "@" **+** label **+** "\n0;JMP\n";

    }

**private** **static** **String** **ifGotoLabel**(**String** ***label***) {

**return** "@SP\nAM=M-1\nD=M\n@" **+** label **+** "\nD;JNE\n";

    }

**public** **static** **void** **main**(**String**[] ***args***) {

**try** {

**BufferedReader** br **=** **new** **BufferedReader**(**new** **FileReader**("vm.vm"));

**BufferedWriter** bw **=** **new** **BufferedWriter**(**new** **FileWriter**("output.asm"));

**String** line;

**while** ((line **=** br.**readLine**()) **!=** null) {

                line **=** line.**trim**();

**if** (line.**isEmpty**() **||** line.**startsWith**("//")) {

**continue**;

                }

**String**[] parts **=** line.**split**("\\s+");

**String** command **=** parts[0];

**String** arg1 **=** null;

**String** arg2 **=** null;

**if** (parts.length **>** 1) {

                    arg1 **=** parts[1];

                }

**if** (parts.length **>** 2) {

                    arg2 **=** parts[2];

                }

**String** asmCode **=** "";

**if** (command.**equals**("push")) {

**if** (arg1.**equals**("constant")) {

                        asmCode **=** **pushConstant**(arg2);

                    } **else** **if** (arg1.**equals**("static")) {

                        asmCode **=** **pushStatic**(arg2);

                    } **else** **if** (arg1.**equals**("pointer")) {

                        asmCode **=** **pushPointer**(arg2);

                    } **else** {

                        asmCode **=** **pushSegment**(arg1, arg2);

                    }

                } **else** **if** (command.**equals**("pop")) {

**if** (arg1.**equals**("static")) {

                        asmCode **=** **popStatic**(arg2);

                    } **else** **if** (arg1.**equals**("pointer")) {

                        asmCode **=** **popPointer**(arg2);

                    } **else** {

                        asmCode **=** **popSegment**(arg1, arg2);

                    }

                } **else** **if** (command.**equals**("label")) {

                    asmCode **=** **label**(arg1);

                } **else** **if** (command.**equals**("goto")) {

                    asmCode **=** **gotoLabel**(arg1);

                } **else** **if** (command.**equals**("if-goto")) {

                    asmCode **=** **ifGotoLabel**(arg1);

                } **else** {

                    asmCode **=** **arithSegment**(command);

                }

                bw.**write**("// " **+** line **+** "\n");

                bw.**write**(asmCode);

            }

            br.**close**();

            bw.**close**();

            System.out.**println**("Translation completed successfully!");

        } **catch** (**IOException** ***e***) {

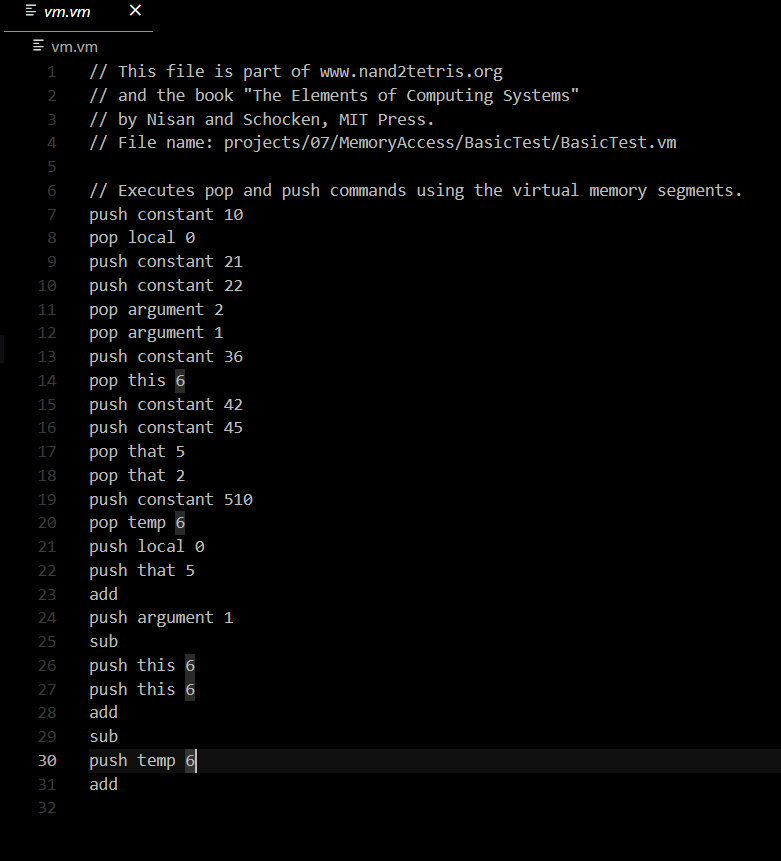
            e.**printStackTrace**();

        }

    }

}

**Input VM file:**

****

**Output asm file:**

**// push constant 10**

**@10**

**D=A**

**@SP**

**A=M**

**M=D**

**@SP**

**M=M+1**

**// pop local 0**

**@0**

**D=A**

**@LCL**

**D=D+M**

**@R13**

**M=D**

**@SP**

**AM=M-1**

**D=M**

**@R13**

**A=M**

**M=D**

**// push constant 21**

**@21**

**D=A**

**@SP**

**A=M**

**M=D**

**@SP**

**M=M+1**

**// push constant 22**

**@22**

**D=A**

**@SP**

**A=M**

**M=D**

**@SP**

**M=M+1**

**// pop argument 2**

**@2**

**D=A**

**@ARG**

**D=D+M**

**@R13**

**M=D**

**@SP**

**AM=M-1**

**D=M**

**@R13**

**A=M**

**M=D**

**// pop argument 1**

**@1**

**D=A**

**@ARG**

**D=D+M**

**@R13**

**M=D**

**@SP**

**AM=M-1**

**D=M**

**@R13**

**A=M**

**M=D**

**// push constant 36**

**@36**

**D=A**

**@SP**

**A=M**

**M=D**

**@SP**

**M=M+1**

**// pop this 6**

**@6**

**D=A**

**@THIS**

**D=D+M**

**@R13**

**M=D**

**@SP**

**AM=M-1**

**D=M**

**@R13**

**A=M**

**M=D**

**// push constant 42**

**@42**

**D=A**

**@SP**

**A=M**

**M=D**

**@SP**

**M=M+1**

**// push constant 45**

**@45**

**D=A**

**@SP**

**A=M**

**M=D**

**@SP**

**M=M+1**

**// pop that 5**

**@5**

**D=A**

**@THAT**

**D=D+M**

**@R13**

**M=D**

**@SP**

**AM=M-1**

**D=M**

**@R13**

**A=M**

**M=D**

**// pop that 2**

**@2**

**D=A**

**@THAT**

**D=D+M**

**@R13**

**M=D**

**@SP**

**AM=M-1**

**D=M**

**@R13**

**A=M**

**M=D**

**// push constant 510**

**@510**

**D=A**

**@SP**

**A=M**

**M=D**

**@SP**

**M=M+1**

**// pop temp 6**

**@6**

**D=A**

**@R5**

**D=D+M**

**@R13**

**M=D**

**@SP**

**AM=M-1**

**D=M**

**@R13**

**A=M**

**M=D**

**// push local 0**

**@0**

**D=A**

**@LCL**

**A=D+M**

**D=M**

**@SP**

**A=M**

**M=D**

**@SP**

**M=M+1**

**// push that 5**

**@5**

**D=A**

**@THAT**

**A=D+M**

**D=M**

**@SP**

**A=M**

**M=D**

**@SP**

**M=M+1**

**// add**

**@SP**

**AM=M-1**

**D=M**

**A=A-1**

**M=M+D**

**// push argument 1**

**@1**

**D=A**

**@ARG**

**A=D+M**

**D=M**

**@SP**

**A=M**

**M=D**

**@SP**

**M=M+1**

**// sub**

**@SP**

**AM=M-1**

**D=M**

**A=A-1**

**M=M-D**

**// push this 6**

**@6**

**D=A**

**@THIS**

**A=D+M**

**D=M**

**@SP**

**A=M**

**M=D**

**@SP**

**M=M+1**

**// push this 6**

**@6**

**D=A**

**@THIS**

**A=D+M**

**D=M**

**@SP**

**A=M**

**M=D**

**@SP**

**M=M+1**

**// add**

**@SP**

**AM=M-1**

**D=M**

**A=A-1**

**M=M+D**

**// sub**

**@SP**

**AM=M-1**

**D=M**

**A=A-1**

**M=M-D**

**// push temp 6**

**@6**

**D=A**

**@R5**

**A=D+M**

**D=M**

**@SP**

**A=M**

**M=D**

**@SP**

**M=M+1**

**// add**

**@SP**

**AM=M-1**

**D=M**

**A=A-1**

**M=M+D**