BL.EN.U4AIE19027 K. SAINIKHILESH REDDY

**END-SEMESTER**

**PROJECT NAME : HACK ASSEMBLER**

**BACKGROUND:-**

Low-level machine programs are rarely written by humans. Typically, they are generated by compilers. Yet humans can inspect the translated code and learn important lessons about how to write their high-level programs better, in a way that avoids low-level pitfalls and exploits the underlying hardware better. One of the key players in this translation process is the assembler -- a program designed to translate code written in a symbolic machine language into code written in binary machine language.

​

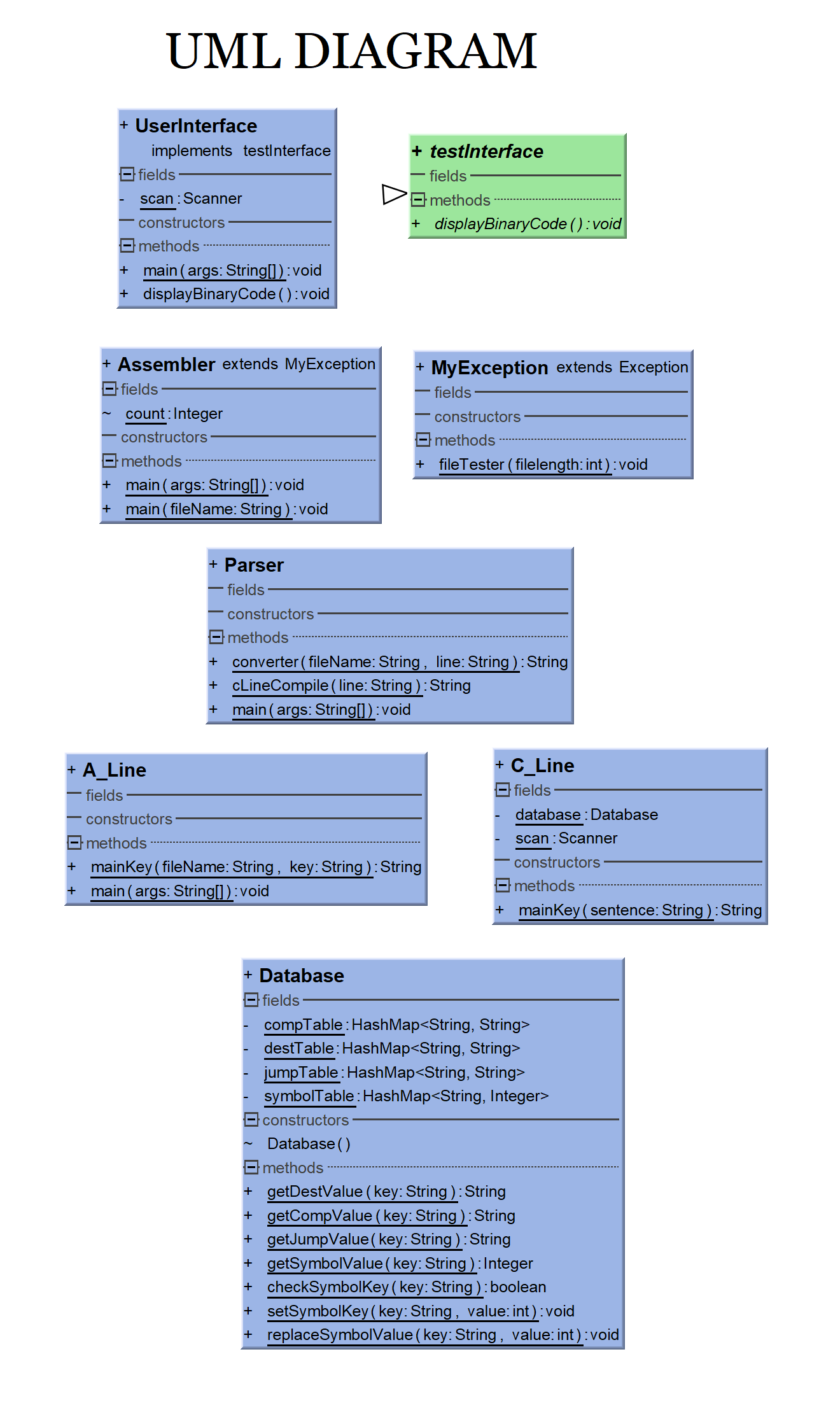
​ it deals with building the first rung up the software hierarchy, which will eventually end up in the construction of a compiler for a Java-like high-level language. But, first things first.

**OBJECTIVE:-**

Write an Assembler program that translates programs written in the symbolic Hack assembly language into binary code that can execute on the Hack hardware platform

**USAGE:-**

Depending on the programming language that you use, the assembler should be invoked using something like "Assembler fileName.asm", where the string fileName.asm is the assembler's input, i.e. the name of a text file containing Hack assembly commands. The assembler creates an output text file named fileName.hack. Each line in the output file consists of sixteen 0 and 1 characters. The output file is stored in the same directory of the input file. The name of the input file may contain a file path. (or we can run this in interactive mode by using “java UserInterface”)



Code:- // Database.java

import java.util.HashMap;

/\*\*

\* Created by Sai Nikhilesh Reddy at 16:33 , on 20-12-2019.

\*/

public class Database {

private static HashMap<String,String> compTable = new HashMap<>();

private static HashMap<String ,String> destTable = new HashMap<>();

private static HashMap<String,String> jumpTable = new HashMap<>();

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

Database(){

// COMPTABLE a=0

compTable.put("0","0101010"); compTable.put("1","0111111"); compTable.put("-1","0111010");

compTable.put("D","0001100"); compTable.put("A","0110000"); compTable.put("!D","0001101");

compTable.put("!A","0110001"); compTable.put("-D","0001111"); compTable.put("-A","0110011");

compTable.put("D+1","0011111"); compTable.put("A+1","0110111"); compTable.put("D-1","0001110");

compTable.put("A-1","0110010"); compTable.put("D+A","0000010"); compTable.put("D-A","0010011");

compTable.put("A-D","0000111"); compTable.put("D&A","0000000"); compTable.put("D|A","0010101");

// COMPTABLE a=1

compTable.put("M","1110000"); compTable.put("!M","1110001"); compTable.put("-M","1110011");

compTable.put("M+1","1110111"); compTable.put("M-1","1110010"); compTable.put("D+M","1000010");

compTable.put("D-M","1010011"); compTable.put("M-D","1000111"); compTable.put("D&M","1000000");

compTable.put("D|M","1010101");

// DESTTABLE

destTable.put("","000"); destTable.put("M","001"); destTable.put("D","010");

destTable.put("MD","011"); destTable.put("A","100"); destTable.put("AM","101");

destTable.put("AD","110"); destTable.put("AMD","111");

// JUMPTABLE

jumpTable.put("","000"); jumpTable.put("JGT","001"); jumpTable.put("JEQ","010");

jumpTable.put("JGE","011"); jumpTable.put("JLT","100"); jumpTable.put("JNE","101");

jumpTable.put("JLE","110"); jumpTable.put("JMP","111");

// SYMBOLTABLE

symbolTable.put("R0",0); symbolTable.put("R1",1); symbolTable.put("R2",2);

symbolTable.put("R3",3); symbolTable.put("R4",4); symbolTable.put("R5",5);

symbolTable.put("R6",6); symbolTable.put("R7",7); symbolTable.put("R8",8);

symbolTable.put("R9",9); symbolTable.put("R10",10); symbolTable.put("R11",11);

symbolTable.put("R12",12); symbolTable.put("R13",13); symbolTable.put("R14",14);

symbolTable.put("R15",15); symbolTable.put("SCREEN",16384); symbolTable.put("KBD",24576);

symbolTable.put("0",0); symbolTable.put("1",1); symbolTable.put("2",2);

symbolTable.put("3",3); symbolTable.put("4",4); symbolTable.put("5",5);

symbolTable.put("6",6); symbolTable.put("7",7); symbolTable.put("8",8);

symbolTable.put("9",9); symbolTable.put("10",10); symbolTable.put("11",11);

symbolTable.put("12",12); symbolTable.put("13",13); symbolTable.put("14",14);

symbolTable.put("15",15);

// SYMBOLTABLE --- SPECIAL KEYS

symbolTable.put("SP",0); symbolTable.put("LCL",1); symbolTable.put("ARG",2);

symbolTable.put("THIS",3); symbolTable.put("THAT",4);

symbolTable.replace("count",15);

}

public static String getDestValue(String key){

if (destTable.containsKey(key)){

return destTable.get(key);

}

return null;

}

public static String getCompValue(String key){

if (compTable.containsKey(key)){

return compTable.get(key);

}

return null;

}

public static String getJumpValue(String key){

if (jumpTable.containsKey(key)){

return jumpTable.get(key);

}

return null;

}

public static Integer getSymbolValue(String key){

if (symbolTable.containsKey(key)){

return symbolTable.get(key);

}

return null;

}

public static boolean checkSymbolKey(String key){

return (symbolTable.containsKey(key));

}

public static void setSymbolKey(String key, int value){

symbolTable.put(key,value);

}

public static void replaceSymbolValue(String key,int value){

symbolTable.replace(key,value);

}

}

// Assembler.java

import java.io.File;

import java.io.FileNotFoundException;

import java.io.PrintStream;

import java.util.Scanner;

/\*\*

\* Created by Sai Nikhilesh Reddy at 12:29 , on 22-12-2019.

\*/

public class Assembler extends MyException{

static Integer count = 0;

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

try{

fileTester(args[0].length());

String fileName = args[0]; //args[0];

String outputFile = fileName.substring(0,fileName.indexOf('.'))+".hack";

PrintStream ps = new PrintStream(new File(outputFile));

Scanner data = new Scanner(new File(fileName));

while(data.hasNextLine()){

String binaryLine = (Parser.converter(fileName, data.nextLine()));

if (binaryLine != null)

ps.println(binaryLine);

}

}catch(ArrayIndexOutOfBoundsException e){

System.out.println("There's no input for file.");

}catch(FileNotFoundException e){

System.out.println("File not Found");

}

}

public static void main(String fileName){

try{

Scanner data = new Scanner(new File(fileName));

String outputFile = fileName.substring(0,fileName.indexOf('.'))+".hack";

PrintStream ps = new PrintStream(new File(outputFile));

while(data.hasNextLine()){

String binaryLine = (Parser.converter(fileName, data.nextLine()));

if (binaryLine != null)

ps.println(binaryLine);

}

Scanner dataOut = new Scanner(new File(outputFile));

if (dataOut.hasNextLine()){

System.out.println("Successfully writen to "+outputFile);

}else{

System.out.println("Error in processing the file.check for any errors.");

}

}catch(ArrayIndexOutOfBoundsException e){

System.out.println("There's no input for file.");

}catch(FileNotFoundException e){

System.out.println("File not Found");

}

}

}

// MyException.java

public class MyException extends Exception {

public static void fileTester(int filelength) throws ArrayIndexOutOfBoundsException{

if (filelength == 0){

throw(new ArrayIndexOutOfBoundsException("File Not Entered"));

}

}

}

// Parser.java

import java.io.FileNotFoundException;

/\*\*

\* Created by Sai Nikhilesh Reddy at 12:29 , on 22-12-2019.

\*/

public class Parser {

public static String converter(String fileName, String line) throws FileNotFoundException {

int slash = 0;

String lineValue;

char[] lineChar = line.toCharArray();

if (!(lineChar.length < 2)) {

for (int i = 0; i < lineChar.length - 1; i++) {

if (lineChar[i] == '/' && lineChar[i + 1] == '/') {

slash = i;

break;

}

}

}

// System.out.println("Line : "+line);

if (!((line.substring(0, (slash)).trim().isEmpty() && slash != 0) || line.trim().isEmpty()

|| (line.charAt(0) == '/' && line.charAt(1) == '/'))) {

int endPosition = (slash == 0) ? line.length() : slash;

String editedKey = line.substring(0, endPosition).trim();

if ((line.trim().charAt(0) == '@') || (line.trim().charAt(0) == '(')) {

lineValue = A\_Line.mainKey(fileName, editedKey);

return lineValue;

} else {

lineValue = cLineCompile(line.trim());

return lineValue;

}

}else{

return null;

}

}

public static String cLineCompile(String line) {

return C\_Line.mainKey(line);

}

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

}

}

// A\_Line.java

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

/\*\*

\* Created by Sai Nikhilesh Reddy at 16:08 , on 22-12-2019.

\*/

public class A\_Line {

public static String mainKey(String fileName, String key) throws FileNotFoundException {

int countHidden = 0;

int countRegister = 15;

int slashHidden = 0;

String line;

Database database = new Database();

Scanner data = new Scanner(new File(fileName));

while (data.hasNextLine()) {

line = data.nextLine();

char[] lineChar = line.toCharArray();

if (!(lineChar.length < 2)) {

for (int i = 0; i < lineChar.length - 1; i++) {

if (lineChar[i] == '/' && lineChar[i + 1] == '/') {

slashHidden = i;

break;

}

}

}

if (!((line.substring(0, (slashHidden)).trim().isEmpty() && slashHidden != 0) || line.trim().isEmpty()

|| (line.charAt(0) == '/' && line.charAt(1) == '/'))) {

int endPosition = (slashHidden == 0) ? line.length() : slashHidden;

String mainKey = line.substring(0, endPosition).trim();

int positionOne = mainKey.indexOf('(');

int positionTwo = mainKey.indexOf(')');

if (positionOne >= 0 && positionTwo > 0) {

String getLabel = mainKey.substring(positionOne + 1, positionTwo).trim();

database.setSymbolKey(getLabel, countHidden);

// System.out.println(getLabel+" ==> "+database.getSymbolValue(getLabel));

}else {

countHidden++;

}

if (mainKey.charAt(0) == '@') {

String getLabel = mainKey.substring(1).trim();

if (!database.checkSymbolKey(getLabel)){

countRegister++;

database.setSymbolKey(getLabel,countRegister);

// System.out.println(getLabel+" ==> "+database.getSymbolValue(getLabel));

}

}

// System.out.println("Line :"+(countHidden-1)+":"+line);

}

slashHidden = 0;

}

char[] lineChar = key.toCharArray();

if (!(lineChar.length < 2)) {

for (int i = 0; i < lineChar.length - 1; i++) {

if (lineChar[i] == '/' && lineChar[i + 1] == '/') {

slashHidden = i;

break;

}

}

}

// System.out.println("slash " + slashHidden);

String shortString = null;

if (key.charAt(0) == '@') {

int endPosition = (slashHidden == 0) ? key.length() : slashHidden;

shortString = key.substring(1, endPosition).trim();

// System.out.println(shortString+"==>"+database.getSymbolValue(shortString));

return (String.format("%16s", Integer.toBinaryString(database.getSymbolValue(shortString)))

.replace(' ', '0'));

}

/\*if (key.charAt(0) == '('){

shortString = key.substring(1, key.indexOf(')'));

// System.out.println(shortString);

// System.out.println(shortString+"==>"+database.getSymbolValue(shortString));

}\*/

return shortString;

}

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

}

}

// C\_Line.java

import java.util.Scanner;

/\*\*

\* Created by Sai Nikhilesh Reddy at 10:36 , on 18-12-2019.

\*/

public class C\_Line {

private static Database database = new Database();

private static Scanner scan = new Scanner(System.in);

public static String mainKey(String sentence){

String cLine = sentence;

int pDest = 0;

String sDest = null;

String destValue;

int pComp = 0;

String sComp = null;

String compValue;

int pJump = 0;

String sJump = null;

String jumpValue;

int commentVariablePositon = 0 ;

String finalMachineCode;

for (int i = 0; i<cLine.length(); i++){

// comment variable position

if (i<=(cLine.length()-2)){

if (cLine.charAt(i) == '/' && cLine.charAt(i+1) == '/'){

commentVariablePositon = i;

}

}

// dest KEYWORD

if (cLine.charAt(i) == '=' && pDest ==0){

if (i<commentVariablePositon || commentVariablePositon==0){

pDest = i;

sDest = cLine.substring(0,(i)).trim();

}

}

// comp KEYWORD

if (cLine.charAt(i) == ';' && pComp == 0){

pComp = i;

}

if (pDest !=0 && pComp ==0){

int positionTemp = (commentVariablePositon == 0) ? cLine.length() : commentVariablePositon;

sComp = cLine.substring((pDest+1),(positionTemp)).trim();

}

if (pDest==0 && pComp !=0){

sComp = cLine.substring(pDest,pComp);

}

// jump KEYWORD

if (pComp != 0 && pDest == 0){

pJump = (commentVariablePositon == 0) ? cLine.length() : commentVariablePositon;

sJump = cLine.substring((pComp+1),pJump).trim();

}

}

String value = null;

value = (sDest == null) ? "" : sDest;

destValue = database.getDestValue(value);

compValue = database.getCompValue(sComp);

value = (sJump == null) ? "" : sJump;

jumpValue = database.getJumpValue(value);

// System.out.println("Dest key = "+sDest+":"+destValue);

// System.out.println("Comp key = "+sComp+":"+compValue);

// System.out.println("Jump key = "+sJump+":"+jumpValue);

finalMachineCode = "111"+compValue+destValue+jumpValue;

/\*System.out.println("dest is :"+sDest+":");

System.out.println("comp is :"+sComp+":");

System.out.println("jump is :"+sJump+":");

System.out.println("\n\ndest Key :"+sDest+":");

System.out.println("dest Value :"+destValue+":");

System.out.println("comp Key :"+sComp+":");

System.out.println("comp Value :"+compValue+":");

System.out.println("jump Key :"+sJump+":");

System.out.println("jump Value :"+jumpValue+":");

System.out.println("\nFinal code :"+finalMachineCode);\*/

return finalMachineCode;

}

}

// UserInterface.java

import java.io.File;

import java.io.FileNotFoundException;

import java.util.Scanner;

/\*\*

\* Created by Sai Nikhilesh Reddy at 01:15 , on 23-12-2019.

\*/

public class UserInterface implements testInterface{

private static Scanner scan = new Scanner(System.in);

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

System.out.println("==============Hack Assembler==============\n");

while(true){

System.out.print("\nEnter file name : ");

String inputFile = scan.nextLine();

String userFileCheck = inputFile.substring(inputFile.indexOf('.'));

if (userFileCheck.equals(".asm")){

Assembler.main(inputFile);

}else{

System.out.println("Incorrect file format.");

return;

}

String outputFile = inputFile.substring(0,inputFile.indexOf('.')) + ".hack";

System.out.print("\nDo you want to see the contents of "+outputFile+" :");

if (scan.nextLine().toLowerCase().equals("yes")){

Scanner data = new Scanner(new File(outputFile));

while(data.hasNextLine()){

System.out.println(data.nextLine());

}

}

System.out.print("\nDo you want to convert another file :");

if (!scan.nextLine().toLowerCase().equals("yes")){

System.out.println("Thanks for using HACK ASSEMBLER");

break;

}

}

}

@Override

public void displayBinaryCode() {

}

}

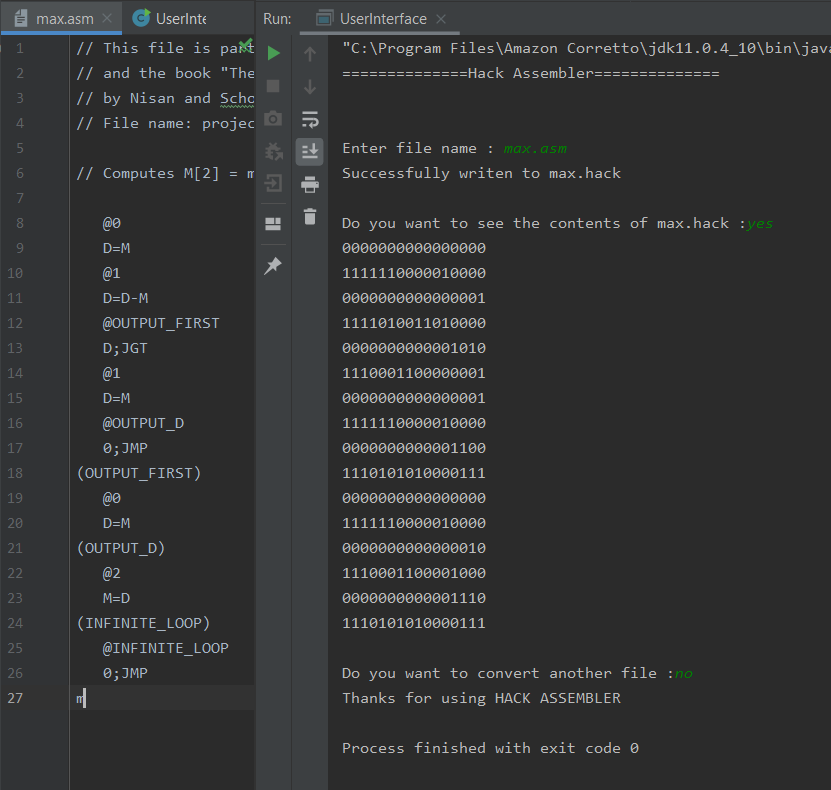
// testInterface.java

public interface testInterface {

void displayBinaryCode();

}

OUTPUT:-



OUTPUT VERIFICATION (USING NAND2TETRIS INBUILT ASSEMBLER):-

