**package** com.Assassin;  
**import** java.util.\*;  
**import** org.apache.commons.collections4.bidimap.DualHashBidiMap;  
**public class** Main {  
  
 **public static void** main(String[] args) {  
 *//long startTime = System.nanoTime();* Scanner input =**new** Scanner(System.in);  
 System.out.println(**"Enter Instruction (space after comma between registers) or opcode: "**);  
 DualHashBidiMap<String,String> registers = **new** DualHashBidiMap<String, String>(){{  
 put(**"A"**,**"0"**);  
 put(**"X"**,**"1"**);  
 put(**"L"**,**"2"**);  
 put(**"B"**,**"3"**);  
 put(**"S"**,**"4"**);  
 put(**"T"**,**"5"**);  
 }};  
 DualHashBidiMap<String,String> instructions = **new** DualHashBidiMap<String, String>(){{  
 put(**"ADDR"**,**"90"**);  
 put(**"CLEAR"**,**"B4"**);  
 put(**"COMPR"**,**"A0"**);  
 put(**"DIVR"**,**"9C"**);  
 put(**"MULR"**,**"98"**);  
 put(**"RMO"**,**"AC"**);  
 put(**"SUBR"**,**"94"**);  
 put(**"TIXR"**,**"B8"**);  
 put(**"SHIFTL"**,**"A4"**);  
 put(**"SHIFTR"**,**"A8"**);  
 put(**"SVC"**,**"B0"**);  
 }};  
 String instruction=input.nextLine().toUpperCase();  
 String[] instSplit=instruction.split(**"\\s+"**);  
 StringBuilder out=**new** StringBuilder();  
 **for**(String i:instSplit){  
 **if**(instructions.containsKey(i)){  
 out.append(instructions.get(i));  
 }  
 **else if**(i.endsWith(**","**)) {  
 out.append(registers.get(i.replace(**","**, **""**)));  
 }  
 **else if**(registers.containsKey(i)) {  
 out.append(registers.get(i));  
 }  
 **else if**(i.matches(**"-?([0-9]\\d\*)"**)){  
 out.append(i);  
 }  
 }  
  
 **if**(instructions.containsValue(instruction.substring(0,2))){  
 **if**(instructions.getKey(instruction.substring(0,2)).equals(**"SHIFTL"**)||instructions.getKey(instruction.substring(0,2)).equals(**"SHIFTR"**)) {  
 out.append(instructions.getKey(instruction.substring(0, 2)) + **" "**);  
 **if** (registers.containsValue(instruction.substring(2, 3))) {  
 out.append(registers.getKey(instruction.substring(2, 3)) + **", "**);  
 }  
 out.append(instruction.substring(3));  
 }  
 **else if**(instructions.getKey(instruction.substring(0,2)).equals(**"TIXR"**)||  
 instructions.getKey(instruction.substring(0,2)).equals(**"CLEAR"**)  
 ) {  
 out.append(instructions.getKey(instruction.substring(0, 2)) + **" "**);  
 **if** (registers.containsValue(instruction.substring(2, 3))) {  
 out.append(registers.getKey(instruction.substring(2, 3)));  
 }  
 }  
 **else if**(instructions.getKey(instruction.substring(0,2)).equals(**"SVC"**)){  
 out.append(instructions.getKey(instruction.substring(0, 2)) +**" "**);  
 out.append(instruction.substring(2,4));  
 }  
 **else if**(!(instructions.getKey(instruction.substring(0,2)).equals(**"SHIFTL"**)  
 &&instructions.getKey(instruction.substring(0,2)).equals(**"SHIFTR"**)  
 &&instructions.getKey(instruction.substring(0,2)).equals(**"TIX"**)  
 &&instructions.getKey(instruction.substring(0,2)).equals(**"CLEAR"**)  
 &&instructions.getKey(instruction.substring(0,2)).equals(**"SVC"**)  
 )){  
 out.append(instructions.getKey(instruction.substring(0,2))+**" "**);  
 **if** (registers.containsValue(instruction.substring(2,3))){  
 out.append(registers.getKey(instruction.substring(2,3))+**", "**);  
 }  
 **if** (registers.containsValue(instruction.substring(3))){  
 out.append(registers.getKey(instruction.substring(3)));  
 }  
 }  
 }  
 **else if**(out.toString().isEmpty())  
 System.out.println(**"Instruction not found!!"**);  
 System.out.println(out);  
 */\*long endTime = System.nanoTime();  
 long timeElapsed = endTime - startTime;  
 System.out.println("Execution time in nanoseconds : " + timeElapsed);  
  
 System.out.println("Execution time in milliseconds : " +  
 timeElapsed / 1000000);\*/* }  
}