Index

Practical 1

Aim: Write A Program To Generate Tokens From Given Input String.

Code:

```
1.  package CD;
2.
3.  import java.io.DataInputStream;
4.  import java.io.File;
5.  import java.io.FileInputStream;
6.  import java.io.FileNotFoundException;
7.  import java.io.IOException;
8.  import java.util.ArrayList;
9.
10. public class TokenParsing {
11.     static ArrayList<String> operator,keyword,deliminator;
12.     @SuppressWarnings("deprecation")
13.     public static void main(String[] args) throws IOException {
14.         InitializeOperator();
15.         InitializeKeyword();
16.         InitializeDeliminator();
17.
18.         try {
19.             DataInputStream dis = new DataInputStream(
20.                     new FileInputStream(
21.                             new File("E:\\IDE\\Eclipse\\Workspace\\S7\\src\\CD\\Inp
    ut.txt")));
22.
23.             String line;
24.             while((line = dis.readLine())!=null)
25.                 Analyize(line);
26.
27.             dis.close();
28.         } catch (FileNotFoundException ex) {
29.
30.         }
31.     }
32.
33.     @SuppressWarnings("deprecation")
34.     private static void InitializeOperator() {
35.         try {
36.             operator = new ArrayList<>();
37.             DataInputStream dis = new DataInputStream(
38.                     new FileInputStream(
39.                             new File("E:\\IDE\\Eclipse\\Workspace\\S7\\src\\CD\\ope
    rator.txt")));
40.             String line;
41.             while((line = dis.readLine())!=null)
42.                 operator.add(line);
43.
44.             dis.close();
45.
46.         } catch (FileNotFoundException ex) {
47.         } catch (IOException ex) {
48.         }
49.     }
50.
51.     @SuppressWarnings("deprecation")
52.     private static void InitializeKeyword() {
53.         try {
```

```
54.             keyword = new ArrayList<>();
55.             DataInputStream dis = new DataInputStream(
56.                     new FileInputStream(
57.                             new File("E:\\IDE\\Eclipse\\Workspace\\S7\\src\\CD\\Key
    word.txt")));
58.             String line;
59.             while((line = dis.readLine())!=null)
60.                 keyword.add(line);
61.
62.             dis.close();
63.
64.         } catch (FileNotFoundException ex) {
65.         } catch (IOException ex) {
66.         }
67.     }
68.
69.     @SuppressWarnings("deprecation")
70.     private static void InitializeDeliminator() {
71.         try {
72.             deliminator = new ArrayList<>();
73.             DataInputStream dis = new DataInputStream(
74.                     new FileInputStream(
75.                             new File("E:\\IDE\\Eclipse\\Workspace\\S7\\src\\CD\\Del
    iminator.txt")));
76.             String line;
77.             while((line = dis.readLine())!=null)
78.                 deliminator.add(line);
79.
80.             dis.close();
81.
82.         } catch (FileNotFoundException ex) {
83.         } catch (IOException ex) {
84.         }
85.     }
86.
87.     private static void Analyize(String line) {
88.         String[] part = line.split(" ");
89.         int firstOccurance = 0;
90.         for(String word : part){
91.
92.             //check if it is keyword
93.             if(keyword.contains(word)){
94.                 System.out.println(word + " is keyword!");
95.                 continue;
96.             }
97.
98.             //check for deli
99.             for(int i=0; i<deliminator.size();i++){
100.                    firstOccurance = 0;
101.                    String deli = deliminator.get(i);
102.                    int currentOccurance = word.indexOf(deli);
103.
104.                    //If that deli is not present
105.                    if(currentOccurance < 0)
106.                        continue;
107.
108.                    String currentWord = word.substring(firstOccurance,currentOc
    curance);
109.                    firstOccurance = currentOccurance;
110.
111.                    if(keyword.contains(currentWord)){
112.                        System.out.println(currentWord + " is Keyword");
113.                        continue;
114.                    }
115.                    int firstOpOcc = 0;
116.                    for(int j=0; j<operator.size();j++){
```

```
                            firstOpOcc=0;
                            String op = operator.get(j);

                            int currentOp = currentWord.indexOf(op);

                            //If that deli is not present
                            if(currentOp < 0)
                                continue;

                            System.out.println(word.substring(firstOpOcc,currentOp)+
    " is variable");
                            firstOpOcc = currentOp+1;

                            if(currentWord.length() == currentOp+1)
                                break;

                            currentWord = currentWord.substring(currentOp+1);
                            j=-1;

                        }

                        if(currentWord.matches("[a-zA-Z]"))
                            System.out.println(currentWord + " is Variable");
                        else
                            System.out.println(currentWord + " is Constant");

                        if(word.length() == currentOccurance+1)
                            break;

                        word = word.substring(currentOccurance+1);
                        i=-1;
                    }
                }
            }
        }
```

Output:

```
void is keyword!
main is Keyword
main() is Constant
int is keyword!
a is variable
5 is Constant
b is Variable
c is Variable
char is keyword!
z is variable
'c' is Constant
b is variable
b is variable
5 is Constant
c is variable
c is variable
10 is Constant
```

Practical 2

Aim: Write A Program To Check Whether Input String Is Accepted By Given DFA or Not.

Dfa_data.java:

```
1.  package CD;
2.  import java.util.Scanner;
3.
4.  public class dfa_data {
5.      int n,s,table[][];
6.      String str,ter;
7.      Scanner in = new Scanner(System.in);
8.
9.      dfa_data(){
10.         System.out.println("Enter No of column and rows and then Table");
11.         n=in.nextInt();
12.         s=in.nextInt();
13.         table=new int[s][n];
14.         for(int i=0;i<s;i++){
15.             for(int j=0;j<n;j++){
16.                 table[i][j]=in.nextInt();
17.             }
18.         }
19.         System.out.println("Enter String to check followed by terminal symbols");
20.         str=in.next();
21.         ter=in.next();
22.     }
23.     int find(char c){
24.         for(int i=0;i<ter.length();i++)
25.             if(ter.charAt(i)==c)
26.                 return i;
27.         return -1;
28.     }
29.     void check(){
30.         int index=0,row=0,col=0;
31.         for(;str.charAt(index)!='#';index++){
32.             col=find(str.charAt(index));
33.             if(col==-1){
34.                 System.out.println("String is Not Valid");
35.                 return;
36.             }
37.             row=table[row][col];
38.             if(row==999){
39.                 System.out.println("String is Not Valid");
40.                 return;
41.             }
42.         }
43.         if(index!=str.length()-1)
44.             System.out.println("String is Not Valid");
45.         else
46.             System.out.println("String is Valid");
47.     }
48. }
```

Dfa.java:

```
1.  package CD;
2.
3.  import java.util.regex.Matcher;
4.  import java.util.regex.Pattern;
5.
6.  public class dfa {
7.      public static void main(String[] args) {
8.          dfa_data work=new dfa_data();
9.          work.check();
10.
11.     }
12. }
```

Output:

```
Enter No of column and rows and then Table
2 5
1 2
0 3
4 999
0 999
999 2
Enter String to check followed by terminal symbols
01010#
01
String is Valid
```

Practical 3

Aim: Write A Program To Find First Of All Non-Terminals Of Given Grammar.

First_NT.java:

```
1.  package CD;
2.
3.  public class First_NT {
4.
5.      char NT,first;
6.      StringBuilder follow;
7.      int done;
8.
9.      public first_NT() {
10.         NT='-';
11.         follow=new StringBuilder();
12.     }
13.
14.     static int indexof(first_NT arry[],char c,int offset){
15.         for(int i=offset;i<arry.length;i++)
16.             if(arry[i].NT==c)
17.                 return i;
18.         return -1;
19.     }
20.
21.     public static void transfer(int toNT, char from, first_NT arry[]){
22.         int fromNT = -1;
23.         while((fromNT=indexof(arry,from,fromNT+1))!=-1){
24.             arry[toNT].follow.append(arry[fromNT].first);
25.         }
26.     }
27.
28.     public static void transferFollow(int toNT, char from, first_NT arry[]){
29.         int fromNT=indexof(arry,from,0);
30.         arry[toNT].follow=arry[fromNT].follow;
31.     }
32.
33.     public static int checkFlag(first_NT obj[],int start, int index){
34.         for(int i=start;i<index;i++)
35.             if(obj[i].done==0)
36.                 return i;
37.         return -1;
38.     }
39.
40. }
```

First.java:

```
1.  package SP;
2.
3.  import java.io.File;
4.  import java.io.FileNotFoundException;
5.  import java.util.Scanner;
6.
7.  public class First {
8.      static first_NT obj[];
9.      public static void main(String[] args) {
10.         Scanner inp = null;
11.         try {
12.             inp = new Scanner(new File("E:\\IDE\\Eclipse\\Workspace\\S5\\src\\SP\\f
    irst.txt"));
13.         } catch (FileNotFoundException e) {
14.             e.printStackTrace();
15.         }
16.         int index=0;
17.         obj=new first_NT[20];
18.         for(int i=0;i<10;i++)
19.             obj[i]=new first_NT();
20.         while(inp.hasNext()){
21.             String token=inp.next();
22.             if(token.charAt(2)>='a' && token.charAt(2)<='z')
23.                 obj[index].done=1;
24.             else
25.                 obj[index].done=0;
26.             obj[index].NT=token.charAt(0);
27.             obj[index].first=token.charAt(2);
28.             index++;
29.         }
30.         int left=first_NT.checkFlag(obj,0,index),flag = 0;
31.         while(left!=-1){
32.             char element = obj[left].first;
33.             for(int i=0;i<index;i++){
34.                 if(obj[i].NT==element){
35.                     obj[left].first=obj[i].first;
36.                     obj[left].done=1;
37.                     left=first_NT.checkFlag(obj,0,index);
38.                     flag=1;
39.                     break;
40.                 }
41.             }
42.             if(flag!=1)
43.                 left=first_NT.checkFlag(obj,left+1,index);
44.         }
45.
46.         for(int j=0;j<index;j++)
47.             System.out.println("First of "+obj[j].NT+" is "+obj[j].first);

48.     }
49. }
```

Input:

```
A=aA;
B=Db;
C=Bc;
D=bb;
Z=ssZ;
```

Output:

```
First of A is a
First of B is b
First of C is b
First of D is b
First of Z is s
```

Practical 4

Aim: Write a program to remove the Left Recursion from a given grammar.

Recursion.java:

```
1.  package CD;
2.
3.  import java.io.File;
4.  import java.io.FileNotFoundException;
5.  import java.util.Scanner;
6.
7.  public class Recursion {
8.
9.      public static void main(String[] args) {
10.         try {
11.             Scanner in = new Scanner(new File("input_rec.txt"));
12.             String newAns[] = new String[20];
13.             int k=0,hasRecursion;
14.             while(in.hasNext()){
15.                 int doneIndex=0,done[]= new int[20];
16.                 hasRecursion=0;
17.                 String rule = in.nextLine();
18.                 char mainNT = rule.charAt(0);
19.                 String part[] = rule.substring(2).split("/");
20.                 for(int i=0;i<part.length;i++){
21.                     if(mainNT==part[i].charAt(0)){
22.                         newAns[k]=new String();
23.                         newAns[k++]=mainNT+"'="+part[i].substring(1)+mainNT+"'"+"|&
    ";
24.                         hasRecursion=1;
25.                     }
26.                     else{
27.                         if(hasRecursion==1){
28.                             newAns[k]=new String();
29.                             newAns[k++]=mainNT+"="+part[i]+mainNT+"'";
30.                             if(i==part.length-1)
31.                                 hasRecursion=0;
32.                         }
33.                         else{
34.                             newAns[k]=new String();
35.                             newAns[k++]=mainNT+"="+part[i];
36.                             done[doneIndex++]=i;
37.                         }
38.                     }
39.
40.                 }
41.                 if(hasRecursion==1){
42.                     if(doneIndex==0){
43.                         newAns[k]=new String();
44.                         newAns[k++]=mainNT+"=#"+mainNT+"'";
45.                     }
46.                 }
47.             }
48.             for(int i=0;i<k;i++)
49.                 System.out.println(newAns[i]);
50.             in.close();
51.         } catch (FileNotFoundException e) {
52.             e.printStackTrace();
```

```
53.         }
54.     }
55.
56. }
```

Input:

```
Z=Aa
A=Aa/AB/Ab/b
B=Ba/Bb/c
C=Ca
```

Output:

```
Z=Aa
A'=aA'|&
A'=BA'|&
A'=bA'|&
A=bA'
B'=aB'|&
B'=bB'|&
B=cB'
C'=aC'|&
C=#C'
```

Practical 5

Aim: Write a program Implement Shift Reduce Parser for the given Grammar.

ShiftReduce.java:

```
1.  package CD;
2.
3.  import java.util.ArrayList;
4.  import java.util.Scanner;
5.
6.  public class ShiftReduce {
7.
8.      static ArrayList<String> rules;
9.
10.     public static void main(String[] args) {
11.         Scanner in = new Scanner(System.in);
12.         int noRules;
13.
14.         System.out.println("Enter No of Rules, followed by Rules(eg E=E+E)");
15.         noRules = in.nextInt();
16.         rules = new ArrayList<>();
17.         for(int i=0;i<noRules;i++)
18.             rules.add(in.next());
19.
20.         char startSymbol=rules.get(0).charAt(0);
21.         System.out.println("Enter String to check");
22.         String statement = in.next();
23.         int statementIndex = 0;
24.
25.
26.         System.out.println("Stack\tInput\tAction");
27.
28.         StringBuilder stack = new StringBuilder();
29.         stack.append("$");
30.         System.out.println(stack.toString()+"\t"+statement.substring(statementIndex
    )+"\t"+"Init");
31.
32.         try{
33.             while(true){
34.                 if(statementIndex<statement.length()){
35.                     stack.append(statement.charAt(statementIndex++));
36.                     System.out.println(stack.toString()
37.                     +"\t"+statement.substring(statementIndex)+"\t"+"Shift");
38.                 }
39.                 check(stack,statement.substring(statementIndex));
40.
41.                 if(stack.length()==2 && stack.charAt(1)==startSymbol && statementIn
    dex>statement.length())
42.                     break;
43.             }
44.         }catch(Exception e){
45.             System.out.println("String Not Accepted");
46.             in.close();
47.             return;
48.         }
49.         System.out.println("String Accepted");
50.         in.close();
51.
```

```
52.     }
53.
54.     private static void check(StringBuilder stack,String statement) {
55.
56.         for(int i=1;i<stack.length();i++){
57.             String subpart = stack.substring(i);
58.             for(int j=0;j<rules.size();j++){
59.                 String ruleInConsideration = rules.get(j);
60.                 if(subpart.equalsIgnoreCase(ruleInConsideration.substring(2))){
61.                     stack.replace(i, stack.length(), "");
62.                     stack.append(ruleInConsideration.charAt(0));
63.                     System.out.println(stack.toString()
64.                     +"\t"+statement+"\t"+"Reduce by "  ruleInConsideration);
65.                     return;
66.                 }
67.             }
68.         }
69.
70.     }
71. }
```

Output:

```
Enter No of Rules, followed by Rules(eg E=E+E)
4
E=E+E
E=E*E
E=(E)
E=i
Enter String to check
i+i*i
Stack   Input   Action
$       i+i*i   Init
$i      +i*i    Shift
$E      +i*i    Reduce by E=i
$E+     i*i     Shift
$E+i    *i      Shift
$E+E    *i      Reduce by E=i
$E+E*   i       Shift
$E+E*i          Shift
$E+E*E          Reduce by E=i
$E+E            Reduce by E=E*E
$E              Reduce by E=E+E
```

Practical 6

Aim: Write a program to generate Symbol table using Hashing.

Symbol_Table.java:

```
1.  package SP;
2.
3.  public class Symbol_Table {
4.      String type,var,value;
5.      int size,addressSum;
6.      static int lastSize;
7.      public symbol_table() {
8.          type=new String();
9.          var=new String();
10.         value=new String();
11.         size=0;
12.     }
13.     void putsize(String token){
14.         int sum=0;
15.         for(int i=0;i<var.length();i++){
16.            char currentChar = var.toLowerCase().charAt(i);
17.            if(currentChar>='1' && currentChar<='9')
18.                sum+= (var.toLowerCase().charAt(i) - 48) * Math.pow(10, i);
19.            else
20.                sum+= (var.toLowerCase().charAt(i) - 96) * Math.pow(10, i);
21.                }
22.         sum%=10;
23.         addressSum = sum;
24.         if(token.equals("int"))
25.             size=4;
26.         else if(token.equals("float"))
27.             size=4;
28.         else
29.             size=2;
30.     }
31.     void putname(String nm,String ty){
32.         var=nm;
33.         type=ty;
34.         putsize(ty);
35.     }
36.     void putval(String val){
37.         value=val;
38.     }
39.     void display(){
40.         System.out.println(var+"\t"+type+"\t"+size+"\t"+value+"\t"+(addressSum+last
    Size));
41.         lastSize=addressSum+lastSize+size;
42.     }
43. }
```

Symbol.java:

```
1.  package SP;
2.
3.  import java.io.File;
4.  import java.io.FileNotFoundException;
5.  import java.util.Scanner;
6.
7.  public class Symbol{
8.      static int MAX_OBJ=10;
9.      public static void main(String[] args) {
10.         Scanner inp = null;
11.         try {
12.             inp = new Scanner(new File("E:\\IDE\\Eclipse\\Workspace\\S5\\src\\SP\\i
    nput.txt"));
13.         } catch (FileNotFoundException e) {
14.             e.printStackTrace();
15.         }
16.         symbol_table obj[]=new symbol_table[MAX_OBJ];
17.         for(int i=0;i<MAX_OBJ;i++)
18.             obj[i]=new symbol_table();
19.         int index=0;
20.         String type = null;
21.         System.out.println("Name\tType\tsize\tValue\tAddress");
22.         while(inp.hasNext()){
23.             String token;
24.
25.             token=inp.next();
26.             int old_pos=0,del_pos=check(token,0);
27.             if(del_pos==0)
28.                 type=token;
29.             else{
30.                 do{
31.                     StringBuilder var=new StringBuilder();
32.                     for(int i=old_pos;i<del_pos;i++)
33.                         var.append(token.charAt(i));
34.
35.                     obj[index].putname(var.toString(),type);
36.                     if(token.charAt(del_pos)=='='){
37.                         old_pos=del_pos+1;
38.                         del_pos=check(token,old_pos);
39.                         if(del_pos!=0){
40.                             StringBuilder temp = new StringBuilder();
41.                             for(int i=old_pos;i<del_pos;i++)
42.                                 temp.append(token.charAt(i));
43.
44.                             obj[index].putval(temp.toString());
45.                         }
46.                         else{
47.                             System.out.println("Error");
48.                         }
49.                     }
50.                     if(token.charAt(del_pos)==',' || token.charAt(del_pos)==';'){
51.                         index++;
52.                     }
53.                     old_pos=del_pos+1;
54.                     del_pos=check(token,old_pos);
55.                 }while(del_pos!=0);
56.             }
57.         }
58.         for(int i=0;i<index;i++)
59.             obj[i].display();
60.     }
```

```
61.
62.     public static int check(String str,int index){
63.         for(int i=index;i<str.length();i++){
64.             if(str.charAt(i)=='='|| str.charAt(i)==';' || str.charAt(i)==',' || str
    .charAt(i)=='+' || str.charAt(i)=='-
    ' || str.charAt(i)=='*' || str.charAt(i)=='/' || str.charAt(i)=='%')
65.                 return i;
66.         }
67.         return 0;
68.     }
69.
70. }
```

Input:

```
int a,b,c=50;
char c1='c';
float z1;
```

Output:

Name	Type	size	Value	Address
a	int	4		1
b	int	4		7
c	int	4	50	14
c1	char	2	'c'	21
z1	float	4		29

Practical 7

Aim: Write a program to implement LL(1) parser.

LL.java:

```
1.  package CD;
2.
3.  import java.util.ArrayList;
4.  import java.util.Scanner;
5.
6.  public class LL {
7.      public static void main(String[] args) {
8.          Scanner in = new Scanner(System.in);
9.          int noNonTerminal,noTerminal;
10.         ArrayList<String> nonTerminal,terminal;
11.         System.out.println("Enter No of Non Terminal, followed by Space Seprated No
    n Terminal");
12.         noNonTerminal = in.nextInt();
13.         nonTerminal = new ArrayList<>();
14.         for(int i=0;i<noNonTerminal;i++)
15.             nonTerminal.add(in.next());
16.         System.out.println("Enter No of Terminal, followed by Space Seprated Termin
    al");
17.         noTerminal = in.nextInt();
18.         terminal = new ArrayList<>();
19.         for(int i=0;i<noTerminal;i++)
20.             terminal.add(in.next());
21.         terminal.add("$");
22.         noTerminal++;
23.
24.         System.out.println("Enter Table. Enter - for no transaction and enter # to
    represent Null ");
25.         String[][] matrix = new String[noNonTerminal][noTerminal];
26.         //System.out.println(terminal);
27.         for(int i=0;i<noNonTerminal;i++)
28.             for(int j=0;j<noTerminal;j++)
29.                 matrix[i][j] = in.next();
30.
31.
32.         ArrayList<String> stack = new ArrayList<>();
33.         System.out.println("Enter String to check, ending with $");
34.         String statement = in.next();
35.         int statementIndex = 0;
36.         boolean wrong = false;
37.
38.         stack.add(nonTerminal.get(0));
39.
40.         while(!stack.isEmpty()){
41.             String currentNTSymbol = stack.get(stack.size()-1);
42.             stack.remove(stack.size()-1);
43.             String currentTSymbol=statement.charAt(statementIndex)+"";
44.
45.             if(currentNTSymbol.equals("#"))
46.                 continue;
47.
48.             if(terminal.contains(currentNTSymbol)){
49.                 if(currentNTSymbol.equals(currentTSymbol)){
50.                     statementIndex++;
```

```
                    continue;
                }

            }

            int currentNT = nonTerminal.indexOf(currentNTSymbol);
            int currentT = terminal.indexOf(currentTSymbol);

            String value = matrix[currentNT][currentT];
            if(value.equals("-")){
                System.out.println("Wrong String");
                wrong = true;
                break;
            }

            for(int i=value.length()-1;i>=0;i--)
                stack.add(value.charAt(i)+"");

        }
        if(!wrong)
            System.out.println("String Accepted");
        in.close();

    }
}
```

Output:

```
Enter No of Non Terminal, followed by Space Seprated Non Terminal
4
S A B C
Enter No of Terminal, followed by Space Seprated Terminal
2
0 1
Enter Table. Enter - for no transaction and enter # to represent Null
- 1AB #
0C 1AC -
0S - -
- 1 -
Enter String to check, ending with $
1010$
String Accepted
```

Practical 8

Aim: Write a program which generates Quadruple Table for the given postfix String.

Quadruple .java:

```
1.  package SP;
2.
3.  import java.util.Scanner;
4.  import java.util.Stack;
5.
6.  public class Quadruple {
7.      public static void main(String[] args) {
8.          Scanner in = new Scanner(System.in);
9.
10.         String exp=in.next();
11.         Stack<String> symbol = new Stack<>();
12.         //System.out.println("\tMove A,"+exp.charAt(0));
13.         //if(exp.charAt(0)>='a' && exp.charAt(0)<='z')
14.         //symbol.add(exp.charAt(0));
15.         System.out.println("Opeation\tOperand1\tOperand2\tResult");
16.         int resultindex=0;
17.         for(int index=0;index<exp.length();index++){
18.             char c=exp.charAt(index);
19.             if(c>='a' && c<='z')
20.                 symbol.push(""+c);
21.             else{
22.
23.                 String op1=symbol.pop(),op2=symbol.pop();
24.                 switch(c){
25.                 case '+':
26.                     System.out.println("+\t\t"+op2+"\t\t"+op1+"\t\tt"+resultindex);

27.                     symbol.push("t"+resultindex++);
28.                     break;
29.                 case '-':
30.                     System.out.println("-
    \t\t"+op2+"\t\t"+op1+"\t\tt"+resultindex);
31.                     symbol.push("t"+resultindex++);
32.                     break;
33.                 case '*':
34.                     System.out.println("*\t\t"+op2+"\t\t"+op1+"\t\tt"+resultindex);

35.                     symbol.push("t"+resultindex++);
36.                     break;
37.                 case '/':
38.                     System.out.println("/\t\t"+op2+"\t\t"+op1+"\t\tt"+resultindex);

39.                     symbol.push("t"+resultindex++);
40.                     break;
41.                 case '=':
42.                     System.out.println("=\t\tt"+(resultindex-
    1)+"\t\t"+"\t\t"+op1);
43.                     symbol.push("t"+resultindex);
44.                     break;
45.                 default:
46.                     break;
```

```
                }
            }
        }

        //for(int i=0;i<symbol.size();i++)
            //System.out.println(symbol.get(i)+"\t DS ");

        in.close();
    }
}
```

Output:

```
abc*d*+e=
Opeation        Operand1        Operand2        Result
*               b               c               t0
*               t0              d               t1
+               a               t1              t2
=               t2                              e
```