Μάθημα: «Οργάνωση Συστημάτων Υπολογιστών»

Χειμερινό Εξάμηνο 2022-2023

3η Εργασία

Υπεύθυνη Εργασιών: Μαρία Τογαντζή (mst@aueb.gr)

"Evaluation of postfix expression in MIPS32 Assembly"

HP-42S RPN Scientific Calculator

Τη δεκαετία του 1950 προτάθηκε ο **Αντίστροφος Πολωνικός Συμβολισμός** (Reverse Polish Notiation, **RPN**) για να χρησιμοποιηθεί σε υπολογιστικές μηχανές της εποχής. Για την ακρίβεια ο **RPN** ήταν ιδιαίτερα δημοφιλής σε πολλά υπολογιστικά συστήματα μέχρι και τις αρχές της δεκαετίας του 80 καθώς και σε υπολογιστές τσέπης, κυρίως για την ελάχιστη μνήμη που χρησιμοποιούσε.

Η συνήθης μορφή μιας παράστασης ονομάζεται **ενδοθεματική** (**infix**), γιατί οι τελεστές των πράξεων τίθενται μεταξύ των τελεστέων. Η **RPN** μορφή μιας παράστασης ονομάζεται **μεταθεματική** (**postfix**) γιατί οι τελεστές βρίσκονται μετά από τους τελεστέους, για παράδειγμα:

Infix: 5 + ((1 + 2) * 4) – 3

Postfix: 5 1 2 + 4 * + 3 -

Παρατηρήστε ότι στην **postfix** μορφή της παράστασης κάθε τελεστής ακολουθεί τα ορίσματά του. Επίσης δεν χρειάζονται παρενθέσεις.

Ο **υπολογισμός της postfix παράστασης** (evaluation of postfix expression), που αποτελεί το αντικείμενο της παρούσης εργασίας, γίνεται μέσω μιας στοίβας ακεραίων, ως εξής:

- Όταν η είσοδος είναι τελεστέος (ακέραιος) τον τοποθετούμε (push) στη στοίβα.
- Όταν η είσοδος είναι τελεστής (πράξη), αφαιρούμε από τη στοίβα δύο τελεστέους, κάνουμε την πράξη και τοποθετούμε το αποτέλεσμα στη στοίβα

Input	Operation	Stack	Comment
5	Push operand	5	
1	Push operand	5, 1	
2	Push operand	5, 1, 2	
+	Add	5, 3	Pop two values (1, 2) and push result (3)
4	Push operand	5, 3, 4	
*	Multiply	5, 12	Pop two values (3, 4) and push result (12)
+	Add	17	Pop two values (5, 12) and push result (17)
3	Push operand	17, 3	
−	Subtract	14	Pop two values (17, 3) and push result (14)

Αν η **postfix** παράσταση είναι σωστή, στο τέλος του υπολογισμού, η στοίβα πρέπει να περιέχει μόνο ένα στοιχείο, το αποτέλεσμα. Στο συγκεκριμένο παράδειγμα, το αποτέλεσμα είναι 14.

1. **Ζητείται** να γράψετε ένα πρόγραμμα στη συμβολική γλώσσα του επεξεργαστή **MIPS32** που να διαβάζει μια **postfix** παράσταση και να υπολογίζει την τιμή της.

2. **Θεωρείστε** ότι:

- Ανάμεσα στους όρους της μεταθεματικής παράστασης, ο χρήστης πληκτρολογεί τουλάχιστον έναν κενό χαρακτήρα και στο τέλος το χαρακτήρα '='.
- Οι αριθμοί της παράστασης είναι μη προσημασμένοι ακέραιοι (unsigned int).
- Οι πράξεις που εκτελεί η υπολογιστική μηχανή περιορίζονται στην αριθμητική πρόσθεση, αφαίρεση, πολλαπλασιασμό και διαίρεση (+, -, *, /).

3. **Δίνεται** η περιγραφή της λύσης σε γλώσσα **JAVA**.

Παρατήρηση: στο πρόγραμμα **JAVA,** η υλοποίηση της στοίβας γίνεται μέσω πίνακα. Στο δικό σας πρόγραμμα πρέπει να χρησιμοποιήσετε τη στοίβα του επεξεργαστή μέσω του **καταχωρητή στοίβας ($sp)**. Επίσης πρέπει να υλοποιήσετε τα **υποπρογράμματα**.

```java
/**
 * EvaluatePostfix.java
 *
 * author mst@aueb.gr
 * version 1.00
 */

import java.io.*;
class EvaluatePostfix {

      final static int MAX = 20;
      static int i=0; // stack pointer
      static int [] p = new int [MAX]; // stack


      // SubPrograms

      static int pop () {

            if (i==0) {
                  System.out.println ("Invalid Postfix");
                  System.exit(1);
            }

            i--;

            return (p[i]);

      } // pop


      static void push (int result) {

            p[i] = result;
            i++;

      } //push


      static int calc (int x1, int ch, int x2) {

            int total = 0;

            switch (ch) {
                  case '+'    :     total=x1+x2;
                                    break;
                  case '-'    :     total= x1-x2;
                                    break;
                  case '*'    :     total= x1*x2;
                                    break;
                  case '/'    :     if (x2 != 0) total= x1/x2;
                                    else {
                                       System.out.println("Divide by zero");
                                       System.exit(1);
                                    }
                                    break;
            } // switch

            return total;

      } // calc
```

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

        int result=0,ch, x1, x2, number;

        System.out.print ("Postfix (input): ");

        // While there are input tokens left

        do {
            // Read the next token from input

            ch=(int)System.in.read();

            if (ch != ' ') {

            // If token is a value read an unsigned integer

            number=0;

            while ((ch >='0') && (ch<='9')) {

                number = 10*number + (ch-48);
                ch=(int)System.in.read();
            }

            // If token is operator Pop the top 2 values from the stack
            // Evaluate the operator, with the values as arguments
            // Push the returned results back onto the stack

            if ((ch == '+') || (ch == '-') ||(ch == '*')||(ch == '/')){

                x2= pop ();
                x1= pop ();
                result=calc(x1,ch,x2);
                push (result);
            }

            // If the token is a value Push it onto the stack

            else if (ch != '=')

                        push (number);
            }

        } while (ch != '=');

        // If there is only one value in the stack
        // That value is the result of the calculation

        if (i==1)
            System.out.println("Postfix Evaluation: "+ p[0]);
        else
            // If there are more values in the stack

            System.out.println ("Invalid Postfix");
    } //main

} //EvaluatePostfix
```

Αν και η μετατροπή μιας **infix** παράστασης σε **postfix** δεν αποτελεί αντικείμενο της παρούσης εργασίας, αξίζει να σημειωθεί ότι ο **Edsger W. Dijkstra** επινόησε τον αλγόριθμο **shunting-yard** για να μετατρέψει τις παραστάσεις από infix σε postfix (RPN).

Ο E. Dijkstra (**ACM Turing Award, 1972**), γνωστός μεταξύ άλλων και για τον αλγόριθμο που επιλύει το πρόβλημα της εύρεσης συντομότερων διαδρομών από κοινή αφετηρία (**Dijkstra's Shortest Path Algorithm**), ανακηρύχθηκε το **2001** σε επίτιμο διδάκτορα του Πανεπιστημίου μας μετά από πρόταση του Τμήματος Πληροφορικής.

E.W.Dijkstra Archive: Home page (utexas.edu)

Οδηγίες Παράδοσης:

- Η εργασία είναι ομαδική **2 ατόμων**.
- **Ονομάστε το αρχείο** που περιέχει το πρόγραμμα, με τους αριθμούς του φοιτητικού σας μητρώου και κατάληξη **.s** (για παράδειγμα 3210000_3210010.s).
- Στην αρχή του προγράμματος πρέπει να αναφέρονται σε σχόλιο, **τα ονόματα και οι αριθμοί μητρώου των μελών της ομάδας.**
- **Το πρόγραμμα πρέπει να περιέχει σχόλια**. Τα σχόλια θα πρέπει να είναι γραμμένα με λατινικούς χαρακτήρες (όχι Ελληνικά). Δείτε ως παράδειγμα κατάλληλων σχολίων τα προγράμματα στις διαφάνειες του φροντιστηρίου.
- Εκτελέστε το πρόγραμμά σας στον προσομοιωτή **SPIM** και βεβαιωθείτε ότι δεν έχει συντακτικά ή άλλα σφάλματα.
- **Συμπιέστε** (.zip ή .rar) το αρχείο που περιέχει το πρόγραμμα και **ανεβάστε** το στο eclass μέχρι την αναφερόμενη ημερομηνία. Τη διαδικασία αυτή πρέπει να την κάνει κάθε μέλος της ομάδας, δηλαδή την ίδια εργασία ανεβάζουν στο eclass και τα 2 μέλη της ομάδας. **Προσοχή**: Θα πρέπει τόσο στο όνομα του αρχείου όσο και στα σχόλια να αναφέρονται και οι δύο δημιουργοί της εργασίας
- **Προθεσμία παράδοσης εργασίας**: μέχρι και **13/1/2023**.
- Τέλος, μπορείτε, εφόσον το επιθυμείτε να κάνετε την εργασία και ατομικά.
- Για οποιαδήποτε απορία μπορείτε να επικοινωνείτε με την κ. Μαρία Τογαντζή mst@aueb.gr

Καλή Επιτυχία!