UNIVERSITÀ di VERONA | Dipartimento di INFORMATICA

ELABORATO ASSEMBLY
ARCHITETTURA DEGLI ELABORATI

Università degli Studi di Verona
Corso di laurea in Informatica
Anno scolastico 2020-2021

Elaborato: Reverse Polish Notation-RPN

STUDENTI: Iliescu Michele Eduardo (VR460003) e Pagliarusco Eleonora (VR446508)

STRUTTURA DELL' ELABORATO

Il nostro progetto è composto da:

1)Introduzione

2)Variabili

3)Modalità di passaggio

4)Pseudo-codice in linguaggio C

Spieghiamo la struttura di queste fasi:

1) Introduzione:

La notazione polacca inversa (reverse polish notation, RPN) è una notazione per la scrittura di espressioni aritmetiche in cui gli operatori binari, anziché utilizzare la tradizionale notazione infissa, usano quella postfissa; ad esempio, l'espressione 5 + 2 in RPN verrebbe scritta 5 2 +. La RPN è particolarmente utile perché non necessita dell'utilizzo di parentesi.

Si considerino ad esempio le due espressioni:
- 2 * 5 + 1
- 2 * (5 + 1)

Nel secondo caso le parentesi sono necessarie per indicare che l'addizione va eseguita prima della moltiplicazione.

In RPN questo non è necessario perché le due espressioni vengono scritte in maniera diversa: mentre la prima corrisponde a 2 5 * 1 +, la seconda viene scritta come 2 5 1 + *.

Un altro vantaggio della RPN è quello di essere facilmente implementabile utilizzando uno stack.

Per calcolare il valore di un'espressione, è sufficiente scandirla da sinistra verso destra: quando viene letto un numero lo si salva nello stack, quando viene letta un'operazione binaria si prelevano due numeri dallo stack, si esegue l'operazione tra tali numeri e si salva nuovamente il risultato nello stack.

Ad esempio, volendo valutare il valore dell'espressione 2 5 1 + *, si procede nel seguente modo:
1. metto il valore 2 nello stack;
2. metto il valore 5 nello stack;
3. metto il valore 1 nello stack;
4. estraggo i primi due valori memorizzati in cima allo stack (5 e 1);
5. faccio la somma e salvo il risultato nello stack;
6. estraggo i primi due valori memorizzati in cima allo stack (2 e 6);
7. faccio la moltiplicazione e salvo il risultato;
8. A questo punto l'intera stringa è stata elaborata e nello stack è memorizzato il risultato finale.

Si scriva un programma in assembly che legga in input una stringa rappresentante un'espressione ben formata in numero di operandi e operazioni in RPN (si considerino solo gli operatori + - * /) e scriva in output il risultato ottenuto dalla valutazione dell'espressione.

Si può usare questo calcolatore online per vedere l'esecuzione passo-passo:
Clicca qui per essere indirizzato al sito

2) Variabili:

- *var_invalid* : è la stringa che viene stampata nel file di output nel caso la stringa presente in input non è scritta correttamente in formato RPN

- *var_controllo_operazioni* : è una variabile in cui viene messa a 1 nel caso in cui è stata fatta un'operazione (+ - / *)

- *numero_negativo* : è una variabile che viene messa a 1 quando si è trovato un numero negativo, usata per distinguere un numero negativo da una sottrazione. Rispettivamente al valore della variabile verrà poi effettuata un'addizione o sottrazione in base al formato RPN presente nel file

3)Modalità di passaggio

Il file " main.c " ha una funzione chiamata postfix, in cui riceve come parametri la stringa in formato RPN dal file di input. Dopo di ché in linguaggio assembly verifica che il formato della stringa sia corretta e se lo è farà il calcolo e restituirà un file di output in cui all'interno ci sarà il risultato, se la stringa non fosse scritta correttamente nel file di output ci sarà scritto " Invalid ".

4)Pseudo-codice in linguaggio C

```c
/*global*/char invalid[] = "Invalid";
/*global*/int var_controllo_operazioni;
/*global*/int numero_negativo;

int main(*file_input){
    char input[] = "1 1 + 2 +";
    int result;
    int eax, ebx, ecx, edi;
    char output[];

    int i = 0;
    while (input[i] != '/0') {
        if (input[i] >= 0 && input[i] <= 9) {
            if(numero negativo == 1){
                eax = input[i] * -1;
                i++;
                nemero negativo = 0;
                 /*torno al while*/
            }
            else{
                eax = input[i]; /*push eax nello stack*/
                i++;
                 /*torno al while*/
            }
        }
    /////////
        if(input[i] == '-' ) {
            if(input[i+1] == '/0'){
                eax -= ebx;/*push eax nello stack*/
                result = eax + eax + eax + ..... /*pop tutto dallo stack*/
                output = result;
                return output;
            }

            if(input[i+1] == '/*spazio*/'){
                if(input[i+2] == '/*spazio*/'){
                    output = /*etichetta*/invalid;
                    return output;
                }
                i++; /*torno al while*/
            }
            numero_negativo = 1;
            i++;  /*torno al while*/
        }
    /////////
        if(input[i] == '+' || '*' || '/' ){
            eax += ebx; /*push eax nello stack*/
            i++; /*incremento per muovermi nell'array*/
            /*torno al while*/
        }
    /////////
        if(input[i] <= 0 && input[i] >= 9
            && input[i] != '+'||'*'||'/'||'-') {
            output = /*etichetta*/invalid;
            return output;
        }
    }
    /* ret */
    result = eax + eax + eax + ...../*pop tutto dallo stack*/
                output = result;
                return output;
```