



UNIVERSITÀ DEGLI STUDI DI VERONA

Elaborato ASM

Laboratorio di Architettura degli elaboratori
2020/2021

PROGETTO: Notazione RPN

Gruppo di lavoro:

Stefano Zanolli (VR456008)

DESCRIZIONE PROGETTO

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.

Esempio:


$$2 * (5 + 1) \longrightarrow 2 \ 5 \ 1 \ + \ *$$



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.

VARIABILI UTILIZZATE

r_ eax		r_ = remember Sono le variabili utilizzate per memorizzare i registri più importanti iniziali per poi ripristinarli finito il programma
r_ ebx		
r_ ecx		
r_ edx		
r_ esp		
r_ ebp		

cost —> costante con valore 10 utilizzata per operazioni
come ‘mul’ che non accettano valori immediati (\$)

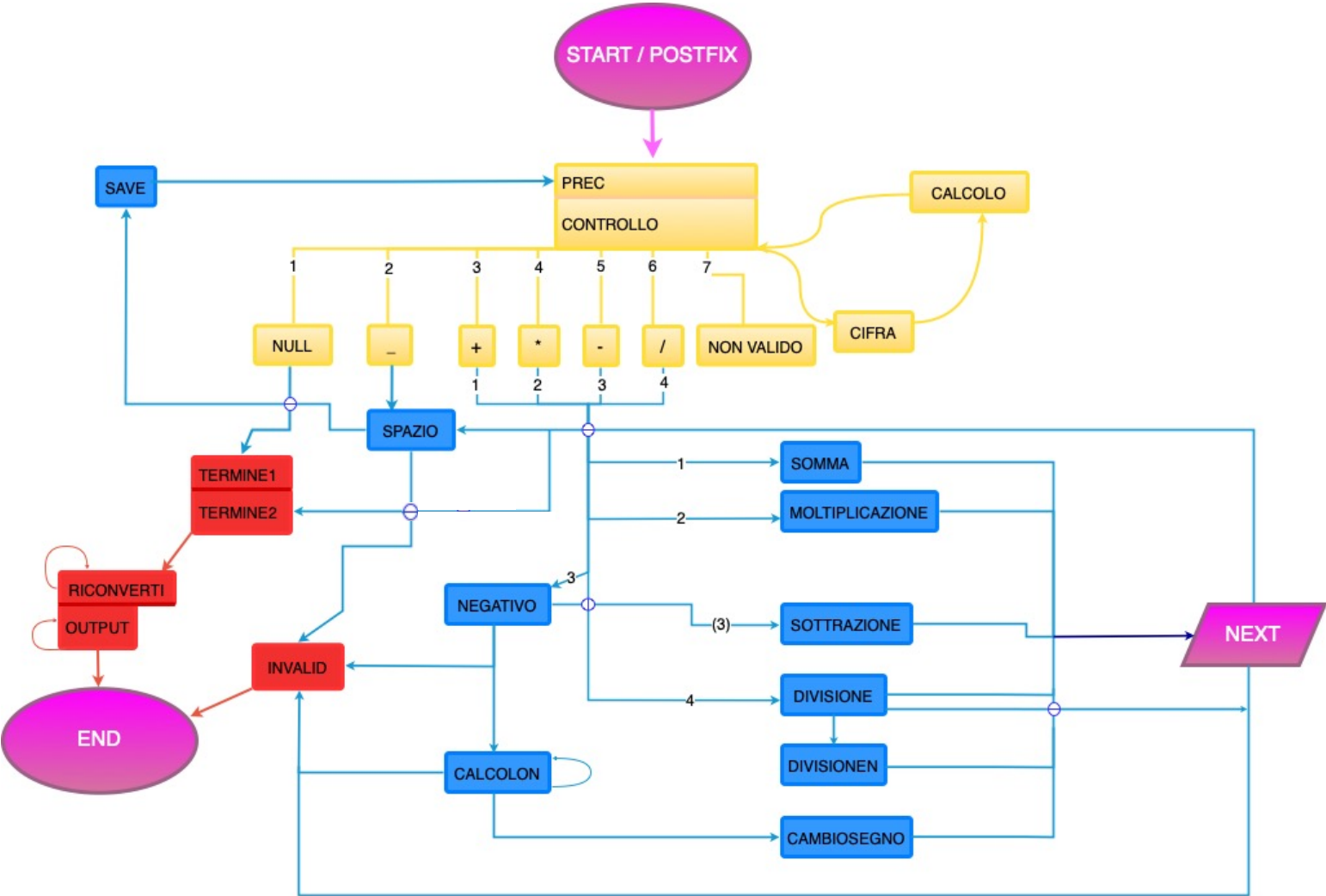
FUNZIONI (Di struttura Di inizio Centrali Di fine)

- **postfix**: è la funzione principale del file assembly ed è richiamata dal file main.c .
 - memorizza i registri iniziali
 - assegna sia l’input che l’output
 - Se l’input è vuoto mette in output “Invalid”
- **Prec**: resetta i registri utili
- **Controllo**: guarda se l’elemento preso in considerazione del vettore in input è un
 - Carattere nullo di fine vettore

- Carattere “spazio”
 - Carattere “somma”
 - Carattere “moltiplicazione”
 - Carattere “segno negativo”
 - Carattere “divisione”
 - Carattere non valido
 - Altrimenti
- **Calcolo:** trasforma l’elemento da ascii ad intero e in caso di più cifre le unisce per formare il valore
 - **Spazio:** si entra in questa etichetta quando si incontra uno “spazio”, controlla il carattere successivo e fa saltare il programma a diverse etichette a seconda se è un
 - Carattere “spazio”
 - Carattere NULL
 - Altrimenti
 - **Somma:** esegue la somma
 - **Moltiplicazione:** esegue la moltiplicazione
 - **Negativo:** guarda l’elemento successivo al segno negativo e salta a diverse etichette a seconda se è un
 - Carattere “spazio”
 - Carattere NULL
 - Carattere invalido
 - Altrimenti
 - **Sottrazione:** esegue la sottrazione
 - **CalcoloN:** trasforma l’elemento da ascii ad intero NEGATIVO e in caso di più cifre le unisce per formare il valore

- **Divisione:** esegue la divisione facendo attenzione che il numeratore sia positivo
- **DivisioneN:** esegue la divisione quando il denominatore è negativo
- **Cambiasegno:** trasforma un intero positivo in negativo
- **Next:** esegue dei controlli a circa metà programma per indirizzarlo all'inizio, all'etichetta Invalid oppure verso la fine
- **Invalid:** mette sul vettore di output la stringa "Invalid"
- **Termine1:** si occupa di prendere il valore finale dallo stack
- **Termine2:** prepara il programma per la successiva etichetta a seconda che il valore finale di eax sia positivo o negativo
- **Riconverti:** riconverte l'elemento in ascii
- **Output:** prende l'elemento riconvertito in ascii e lo carica sul vettore di output
- **END:** ripristina i valori iniziali dei vari registri e con il comando "ret" fa concludere il programma ritornando al file C

FLUSSO



SCELTE PROGETTUALI

1. I valori dei registri iniziali eax,ebx,ecx,edx,esp,ebp sono stati salvati in delle variabili per comodità al posto di inserirli nello stack, dato che la notazione polacca inversa si basa sul continuo utilizzo degli elementi inseriti in quest'ultimo
2. Per leggere numeri più grandi di una singola cifra:
 - prendo la prima cifra e la salvo in eax (che era zero)
 - prendo la seconda cifra e la salvo in eax ma sta volta sommata all'(eax precedente * 10); in questo modo è come se aggiungessi ogni nuova cifra, a destra di eaxesempio: 21 \longrightarrow 2 , eax = 2
1 , eax = 1 + (2 * 10) = 21
3. È stato considerato il carattere “NULL” come simbolo di fine stringa dei vettore di input e output.
4. Divisioni con numeratori negativi danno INVALID
5. L'input non inizia mai con uno spazio ne finisce con un numero non utilizzato