Laboratorio di Architettura degli Elaboratori

Elaborato Assembly

A.A. 2020/2021

Descrizione:

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.

Potete usare questo calcolatore online per vedere l'esecuzione passo-passo:

https://www.free-online-calculator-use.com/postfix-evaluator.html#

Requisiti & Vincoli:

Le espressioni che verranno usate per testare il progetto hanno i seguenti vincoli:

- Gli operatori considerati sono i 4 fondamentali e codificati con i seguenti simboli:
  - + Addizione
  - * Moltiplicazione (non x)
  - - Sottrazione
  - / Divisione (non \)

- Un operando può essere composto da più cifre intere con segno (10, -327, 5670)
- Solo gli operandi negativi hanno il segno riportato esplicitamente in testa.
- Gli operandi hanno un valore massimo codificabile in 32-bit.
- Il risultato di una moltiplicazione o di una divisione può essere codificato al massimo in 32-bit.
- Il risultato di una divisione dà sempre risultati interi, quindi senza resto.
- Il dividendo di una divisione è sempre positivo, mentre il divisore può essere negativo.
- Tra ogni operatore e/o operando vi è uno spazio che li separa.
- L'ultimo operatore dell'espressione è seguito dal simbolo di fine stringa "\0".
- Non è consentito l'utilizzo di chiamate a funzioni descritte in altri linguaggi all'interno del codice Assembly.

Se le stringhe inserite non sono valide (contengono simboli che non sono operatori o numeri) il programma deve restituire la stringa scritta esattamente nel seguente modo: `Invalid`

NON verranno fatti test con stringhe il cui numero di operandi non coincide con il numero di operatori.

Il programma deve essere lanciato da riga di comando con due stringhe come parametri, la prima stringa identifica il nome del file .txt da usare come input, la seconda quello da usare come output:

```
$ ./postfix testin.txt testout.txt
```

Il programma deve leggere il contenuto di `testin.txt` e restituire il risultato della espressione in `testout.txt.`

Il file `testin.txt` contiene una sola riga con l'espressione in RPN da analizzare ed elaborare.

Il file `testout.txt` dovrà contenere solamente il risultato della espressione in RPN data in input attraverso `testin.txt.`

Assieme al presente documento sono forniti i seguenti files:

- `main.c:` Contenente il sorgente C che chiama la funzione assembly e che non può essere editato. La modifica di tale file esclude automaticamente la valutazione dell'elaborato.
- `testin.txt`: File di esempio. In fase di valutazione l'elaborato verrà testato con diverse istanze di diversa complessità.
- `trueout.txt` da utilizzare per verifica del corretto funzionamento del programma. Sarà sufficiente usare il comando

  ```
  $ diff testout.txt trueout.txt
  ```

  per vedere eventuali differenze tra i due files (quello generato dal programma e quello corretto).

In fase di correzione, saranno valutati meglio progetti che ottimizzeranno meglio il codice, ovvero programmi che dimostreranno un minore tempo di esecuzione (a parità di hardware). Durante l'esame orale è possibile che venga richiesto di operare delle modifiche al codice sul momento.

Materiale da consegnare:

Il codice e la relazione vanno compressi in un unico file tarball denominato VRXXXXXX_VRXXXXXX.tar.gz, come precedentemente specificato nel progetto SIS.

Devono essere presenti:

- Sorgenti dell'intero progetto:
    a. Il file assembly principale si dovrà chiamare postfix.s
    b. Il file C, contenente il main e la chiamata assembly, dovrà chiamarsi main.c
- Relazione in formato pdf denominata Relazione.pdf, che affronti nel dettaglio almeno i seguenti punti:
    a. le variabili utilizzate e il loro scopo;
    b. le modalità di passaggio/restituzione dei valori delle funzioni create;
    c. il diagramma di flusso o lo pseudo-codice ad alto livello del codice prodotto;
    d. la descrizione delle scelte progettuali effettuate.

Il pacchetto deve contenere un'unica cartella denominata `asm/` la cui struttura dovrà essere la seguente:

- `asm/`
    - `src/`
        - `postfix.s` ed eventuali altri file di altre funzioni assembly
        - `main.c` non editato
    - `obj/`
        - cartella di supporto per la creazione dei file oggetto (vuota)
    - `bin/`
        - cartella dove verrà creato l'eseguibile (vuota)
    - `Makefile`
        - File `Makefile` per la chain di compilazione. File oggetto generati in `obj/` ed eseguibile generato in `bin/`
    - `Relazione.pdf`
        - File della relazione

Per gli studenti che consegnano entrambi gli elaborati:

Verrà aperta una sezione su moodle per la consegna completa (sis+asm).

Il nome del file rimane invariato (VRXXXXXX_VRXXXXXX.tar.gz).

La struttura del file deve essere la seguente:

- sis/
  - Contiene tutte le sottocartelle ed i files come definito da progetto SIS
- asm/
  - Contiene tutte le sottocartelle ed i files come definito da progetto ASM

Modalità di consegna:

Tutto il materiale va consegnato elettronicamente tramite procedura guidata sul sito Moodle del corso. Indicativamente 15 giorni prima della data di consegna sarà attivata una apposita sezione denominata "consegna_ASM_mmmaaaa" (mmm=mese, aaaa=anno); accedendo a quella pagina sarà possibile effettuare l'upload del materiale. La consegna del materiale comporta automaticamente l'iscrizione all'appello orale.

Si ricorda che è possibile effettuare più sottomissioni, ma ogni nuova sottomissione cancella quella precedente. Ogni gruppo deve consegnare una sola volta il materiale, ovvero un solo membro del gruppo deve effettuare la sottomissione!

Note importanti:

1. È possibile effettuare più sottomissioni, ma ogni nuova sottomissione cancella quella precedente.
2. Un solo membro del gruppo deve effettuare la sottomissione.
3. Tutti i componenti del gruppo devono essere iscritti alla pagina Moodle del corso.
4. Non si accettano progetti consegnati via mail e/o dopo la scadenza.
5. I progetti che non soddisfano tutti i requisiti sopraelencati non verranno ammessi all'orale e non verranno valutati.
6. **Tutti i progetti verranno testati automaticamente. Solo i progetti che supereranno i test saranno ammessi alla discussione orale.**
7. I progetti non ammessi potranno essere visionati e discussi al termine della sessione su richiesta degli studenti.