Laboratorio di Architettura degli Elaboratori

A.A. 2019/20

Elaborato Assembly

Si ottimizzi un codice in linguaggio C che controlla un dispositivo per la gestione intelligente di un parcheggio mediante l'uso di Assembly inline. Il parcheggio è suddiviso in 3 settori: i settori A e B hanno 31 posti macchina ciascuno, mentre il settore C ha 24 posti macchina. Al momento dell'ingresso l'utente deve dichiarare in quale settore vuole parcheggiare, analogamente al momento dell'uscita l'utente deve dichiarare da quale settore proviene.

Il parcheggio rimane libero durante la notte, permettendo a tutte le macchine di entrare e uscire a piacimento. La mattina il dispositivo viene attivato manualmente da un operatore che inserisce manualmente il numero di automobili presenti nei tre settori. Nel suo funzionamento autonomo il sistema riceve in ingresso una stringa contenente il tipo di richiesta che può essere **IN** oppure **OUT**, seguito dal settore in cui l'utente vuole parcheggiare (**A**, **B** o **C**). Ad ogni richiesta il dispositivo risponde aprendo una sbarra e aggiornando il conteggio dei posti liberi nei vari settori. Se un utente chiede di occupare un settore già completo il sistema non deve aprire alcuna sbarra. Nel caso in cui la stringa sia corrotta il dispositivo non deve aprire nessuna sbarra.

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:

```
$ ./parking testin.txt testout.txt
```

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

Il file `testin.txt` contiene nelle prime tre righe il numero di automobili presenti in ogni settore (non necessariamente nell'ordine prestabilito) con la sintassi **SETTORE-NPOSTI** (esempio: A-24). A partire dalla quarta, ogni riga rappresenta una richiesta nella forma **TIPO-SETTORE** (esempio: IN-A oppure OUT-C).

Il file `testout.txt` contiene in ogni riga lo stato del Sistema per quanto riguarda le sbarre e l'occupazione dei settori. La sintassi da utilizzare è **SBARRE-NPOSTIA-NPOSTIB-NPOSTIC-LIGHTS**, con il seguente significato:

- **SBARRE**: è una coppia di char che rappresentano l'apertura (O, open) o chiusura (C, closed) delle sbarre di ingresso e uscita rispettivamente (esempio: CO significa che la sbarra in ingresso è chiusa e quella in uscita aperta).
- **NPOSTIx** (dove x può essere A, B o C): sono il numero di posti attualmente occupati nel rispettivo settore. Usare per tutti i settori rappresentazione a due cifre (es. 04 invece che 4).
- **LIGHTS**: è una terna di valori che rappresenta lo stato di accensione di una luce rossa in corrispondenza a ciascun settore per indicare che è pieno (esempio: 110 significa che i settori A e B sono pieni mentre ci sono ancora posti disponibili nel settore C).

Laboratorio di Architettura degli Elaboratori

A.A. 2019/20

Assieme al presente documento sono forniti:

- un file `parking.c` contenente il sorgente C che dovrà essere editato per inserire la parte assembly. **Il file può essere modificato solo nelle parti segnalate tramite commento!** Files contenenti modifiche esterne a queste sezioni saranno penalizzati e potranno comportare anche un voto insufficiente. È possibile utilizzare assembly inline o funzioni assembly richiamate da C, in quest'ultimo caso i files creati devono essere consegnati all'interno della cartella principale.
- un file `testin.txt` di esempio. In fase di valutazione sarà utilizzato un file di test diverso.
- un file `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.

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.

A causa delle restrizioni alla presenza in dipartimento legate al COVID-19, la correzione degli elaborati si baserà esclusivamente sul materiale consegnato. I progetti funzionanti perfettamente riceveranno una votazione che verrà automaticamente accorpata al voto d'esame. Per i progetti che presentino delle criticità sarà organizzata una sessione orale tramite zoom nei giorni successivi alla consegna, e comunque prima del termine della sessione di esame.

Materiale da consegnare:

Si richiede ad ogni gruppo di caricare un singolo archivio **.tgz** denominato `asm_cognome1_nome1_cognome2_nome2.tgz`[1] contenente tutti i files di seguito elencati senza sottocartelle:

1. Il file `parking.c` modificato, contenente la versione definitiva del progetto da usare per i test. Non rinominare il file fornito ma modificarlo opportunamente

[1] Per generare correttamente un file filename.tgz in linux seguire la seguente procedura:
- posizionare tutti i files da comprimere in una cartella
- rinominare la cartella "filename"
- uscire dalla cartella e lanciare il comando `tar cvfz filename.tgz filename`

2. Tutti gli altri files sorgenti necessari al progetto (ad esempio funzioni assembly)
3. `Makefile` per la compilazione (non fornito, ogni gruppo deve scrivere il proprio). Fare attenzione che il file eseguibile dovrà essere nominato `parking` (senza estensione).
4. Un file `Relazione.pdf` con una relazione del progetto che affronti nel dettaglio almeno i seguenti punti:
    - le variabili utilizzate e il loro scopo;
    - le modalità di passaggio/restituzione dei valori delle funzioni create;
    - il diagramma di flusso o lo pseudo-codice ad alto livello del codice prodotto;
    - la descrizione delle scelte progettuali effettuate

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!