CORSO DI ARCHITETTURA
PROGRAMMAZIONE ASSEMBLY: PROGETTO

17 DICEMBRE 2009

CONSEGNA Venerdì 29 Gennaio 2010

Lo studente programmi nel linguaggio Assembly 8088 un micro 8088 che realizza il sistema sotto descritto. Si utilizzino la versione di assembly e l'assemblatore allegato al libro "Andrew S. Tanenbaum. *Architettura dei Calcolatori. Un Approccio Strutturato*. Pearson Prentice Hall, 5 ed., 2006.", e scaricabile dall'indirizzo internet ftp://ftp.cs.vu.nl/pub/evert.

Descrizione del sistema

Un micro 8088, 1MB di memoria e un'interfaccia di I/O sono usati per realizzare sistema di NGN (Next Generation Networking) in grado di gestire flussi dati con QoS (Quality of Service).
Il sistema è un apparato di rete che analizza i pacchetti in ingresso su diverse porte e ne decide la priorità attraverso inserendo opportunamente il pacchetto in una coda ordinata.
Il cuore del sistema è composto da un motore di ordinamento che processa una sequenza di numeri rappresentanti gli indici di qualità dei pacchetti in coda. **L'ordinamento è fatto con l'algoritmo *merge sort*.**

Emulazioni

I file di testo *input.txt* e *output.txt* emulano le code di elaborazione di ingresso/uscita del sistema.
Le chiamate di sistema per manipolare file sono mascherate dalle funzioni di libreria messe a disposizione dall'assemblatore (_READ, _WRITE, _OPEN, ...).

Punti da svolgere

Si svolgano i seguenti punti in ordine dal Punto 1 fino all'ultimo. Si usino nel codice gli stessi nomi delle funzioni indicate. Si indichi nel codice attraverso un breve commento dove ogni punto viene risolto.
Nelle funzioni si usi, come visto a lezione, **lo stack per il passaggio degli operandi** (siano essi valori o puntatori ad aree di memoria) e **lo stack o il registro AX per ritornare il risultato**.
Si noti che gli operandi denotati da **"*" sono dei puntatori (es. vettore*)**.
I punti 1-2-3-4 sono necessari per valutare sufficientemente l'elaborato. I rimanenti punti concorrono al raggiungimento del massimo dei voti.

Punto 1

Il file *input.txt* contiene la sequenza di indici di qualità da ordinare. Sono decimali disposti nel file separati da uno spazio bianco e possono valere tra 0 e 255. La lunghezza della sequenza è data dalla costante DIM_BUFFER. Si assumi DIM_BUFFER=100.
Si crei il vettore *buffer* i cui elementi sono **byte**.
Si implementino le procedure:

stato CARICA (buffer*)

e

stato SALVA (buffer*)

che:
1- ricevono in ingresso il puntatore al vettore *buffer*;
2- rispettivamente: CARICA copia il contenuto di *input.txt* nel vettore *buffer* convertendolo in intero a 8 bit mentre SALVA crea il file *output.txt* e ne copia il contenuto del vettore *buffer* convertendolo in decimale (*input.txt* se già esiste verrà svuotato);
3- restituiscono lo stato di esecuzione (0 in caso di successo, 1 in caso di errore – ad esempio errori nella apertura, chiusura, lettura del file).

Punto 2
Si implementi la funzione:

boolean MINUSEQUAL (v1, v2)

che riceve in ingresso due operandi di tipo interi ad 8 bit e restituisce 1 (true) se V1 è minore o uguale a V2, altrimenti 0 (false).

Punto 3
Fatto il punto 2, si implementi la procedura:

void MERGE (A*, p, q, r)

che riceve in ingresso il vettore $A[0...n-1]$, i cui elementi sono **byte**, e gli interi p, q e r con $0 \leq p \leq q \leq r \leq n-1$. I sottovettori $A[p...q]$ e $A[q+1...r]$ sono **entrambi ordinati**. La funzione riorganizza il sottovettore $A[p...r]$ in modo che sia ordinato, mente non altera gli elementi di A fuori dal sottovettore $A[p...r]$.

(In sostanza è come se la funzione "fondesse" due vettori **ordinati** vett1 di dim1 elementi e vett2 di dim2 elementi nel vettore vettOut, anch'esso ordinato, di dim1+dim2 elementi. Quindi vett1 è $A[p...q]$, dim1=$q-p+1$, vett2 è $A[q+1...r]$, dim2=$r-(q+1)+1$, vettOut è $A[p...r]$, dimOut=dim1+dim2=$r-p+1$).

Si faccia **obbligatoriamente** uso della funzione MINUSEQUAL, sviluppata al Punto 2, per le operazioni di comparazione tra due elementi di A: $A[i] \leq A[j]$.

Punto 4
Fatto il punto 3, si implementi la procedura:

void MERGE_SORT (A*, p, r)

che riceve in ingresso il vettore $A[0...n-1]$, i cui elementi sono **byte**, e gli interi p e r con $0 \leq p \leq r \leq n-1$. La funzione ordina il sottovettore **non ordinato** $A[p...r]$ mente non altera gli elementi di A fuori dal sottovettore $A[p...r]$.

La procedura è chiamata in maniera ricorsiva ed esegue il seguente codice:

```
MERGE_SORT(A, p, r)
{
      if p < r
      {
          q = (p+r)/2
          MERGE_SORT(A, p, q)
```

```
            MERGE_SORT(A, q + 1, r)
            MERGE(A, p, q, r)
        }
    }
```

(In sostanza MERGE_SORT invoca se stessa su due sotto vettori e una volta ordinati li unisce con la MERGE. Le chiamate ricorsive si annidano finché p<r, cioè fintando che il vettore passato in ingresso alla MERGE è formato da più di 1 elemento).

Esempio di esecuzione

A titolo di esempio, i precedenti punti lavorano in questo modo:

Punto 1 *input.txt* vale 7 1 3
buffer = 00000111
00000001
00000011
output.txt vale 7 1 3

Punto2 V1 = 00000111 e V2 = 00000001
la funziona ritorna 0 (false)

Punto3,4 *A* = 00000111
00000001
00000011
dopo l'esecuzione della procedura
A = 00000001
00000011
00000111

Punto 5

Fatti tutti i punti precedenti, si scriva un programma che **facendo uso delle funzioni e procedure** sviluppate prelevi i DIM_BUFFER valori contenuti in *input.txt* con CARICA e, dopo averi trasferiti nel vettore *buffer* e averli ordinati con MERGE_SORT, li copia in *output.txt* con SALVA.

Punto 6 (opzionale)

Alla luce delle limitazioni espressive dell'assembly, ma anche della capacità di ottimizzazione algoritmica che offre, si discuta l'uso di un diverso algoritmo di ordinamento.

NOTE PER LO SVOLGIMENTO DEL COMPITO

- **Quanto non specificato in questo testo fa parte delle scelte del progettista, quindi dello studente!!! Si riportino le scelte, commentandole, nella relazione.**
- Il docente NON riceve gli studenti durante il periodo di svolgimento del compito. Sarà organizzato un **ricevimento collettivo a 1-2 settimane dal lancio** on-line del compito. Data e luogo del ricevimento collettivo saranno pubblicati alla pagina **http://iafelice.web.cs.unibo.it/archi/**
- Al di fuori del ricevimento collettivo il docente non risponde ne di persona ne via email a domande inerenti il compito, siano esse di carattere teorico sulla materia di esame siano esse di carattere interpretativo del testo del compito.
- Le date della discussione orale saranno pubblicate alla pagina **http://iafelice.web.cs.unibo.it/archi/**

NOTE SULLA VALUTAZIONE DEL COMPITO

La valutazione del progetto dipende strettamente dai seguenti vincoli:

- Il progetto deve essere consegnato entro le 24.00 della data indicata nella prima pagina.
- Saranno attribuiti dei punti ai seguenti aspetti generali:
  1) **Presenza di commenti.** Il codice assembly deve essere commentato in modo da agevolare la comprensione. Ogni funzione deve esser preceduta da opportuni commenti che ne chiariscono la funzionalità e gli argomenti. **I progetti privi di commenti NON saranno valutati.**
  2) **Codice funzionante.** Il progetto deve essere assemblabile e funzionante. **Indicare nella relazione quali sono i Punti di questo testo svolti e quale sistema operativo si è usato.**
  3) **Codice ottimizzato.** Il codice disordinato è difficile da verificare. Saranno valutate positivamente scelte progettuali che ottimizzino il codice. L'ottimizzazione deve riguardare anche i commenti: chiari e coincisi.
  4) **Codice strutturato.** I codice deve essere opportunamente strutturato facendo uso di chiamate a funzioni.
- **Relazione**, di max. 4 pagine per facilitare la lettura del codice. Le pagine eccedenti le 4 consentite non verranno considerate. Si consiglia quindi di evitare copertine, indici, glossari e quanto NON-NECESSARIO o di abbellimento. **Riportare nella relazione le scelte progettuali.**
- **Lavoro di gruppo.** Il progetto **deve** esser svolto da gruppi di lavoro, composti da **almeno due persone e al massimo tre**, in maniera collaborativa da tutti i componenti del gruppo. Ogni componente del gruppo deve conoscere le scelte progettuali e i dettagli implementativi. NON esistono eccezioni a questa regola, ed in particolare **NON sono ammessi progetti singoli**.
- **Confronto tra gruppi.** E' utile il confronto tra gruppi. Tuttavia, dovendo ogni gruppo fare le sue scelte progettuali, è fondamentale evitare scelte di massa. Saranno valutati **negativamente** progetti simili tra loro o con simili scelte progettuali.

MODALITA' DI CONSEGNA

Consegna on-line alla pagina: **http://iafelice.web.cs.unibo.it/archi/**