Delucidazioni su Elaborato Assembly

Lo scopo dell'elaborato era quello di creare un "algoritmo di ordinamento" che permettesse di ordinare 10 prodotti secondo determinati campi (in realtà, se ben implementato l'algoritmo è in grado di ordinarne anche di meno o di più).

I prodotti da ordinare sono suddivisi in 4 campi: ID – durata – scadenza – priorità

I campi per i quali ordinarli sono:

1. *Scadenza minore e se uguali Priorità maggiore* -> **Algoritmo EDF**
2. *Priorità maggiore e se uguali Scadenza minore* -> **Algoritmo HPF**.

Il programma è stato suddiviso in 7 file / funzioni che ne permettono il funzionamento:

pianificatore.s: corpo centrale del programma

intestazione.s: stampa a video il menù con la possibilità di scelta

nome.s: si occupa di:

- prelevare l'indirizzo del file da cui acquisire i dati
- eventualmente anche quello su cui scrivere gli output
- trasformare i numeri acquisiti da formato stringa a intero
- controllare il range di tali numeri, inserirli nella pila (lavoriamo si questa per l'ordinamento)
- eventuali messaggi di errore sui file inseriti / dati acquisiti

edf.s e hpf.s: si tratta dei due algoritmi di ordinamento che, lavorando sulla pila e utilizzando un algoritmo di tipo selection sort (una volta trovato il prodotto minimo, questo viene inserito alla base della pila (base che poi aumenta in modo che non venga sovrascritto dal successivo)), ordinano gli elementi dal minore al maggiore nella pila. Questo procedimento continua fino a quando il selettore non avrà selezionato tutti i prodotti. I meccanismi di ordinamento di edf e hpf sono molto simili, cambiano solo i campi utilizzati per individuare il prodotto, in edf viene cercata la scadenza minima ed eventualmente la priorità massima, in hpf viene cercata invece la priorità massima ed eventualmente la scadenza minima.

printOutput.s: si occupa di stampare a video (*kernel*) i risultati dei prodotti

printOutputFile.s: si occupa di stampare su file (precedentemente inserito) i risultati dei prodotti.

Vediamo ora le funzioni nello specifico:

Pianificatore.s

È il corpo del programma, da esso vengono lanciate tutte le altre funzioni, in ordine ci sono: nomefile, menu, edf, hpf, printOutput, printOutputFile.

La funzione effettiva del pianificatore è permettere il passaggio di variabili tra una funzione e le successive, per esempio il numero di righe presenti nel file (quindi il numero di prodotti) oppure passare eventuali indirizzi e "check" per la scrittura su file.

Il pianificatore si occupa anche di mostrare ciclicamente il menu, quindi a seconda della scelta fatta dall'utente quando gli viene mostrata l'intestazione, verrà eseguita l'operazione e se non vorrà uscire, gli verrà ripresentato il menù.

***Accortezza**:

//La variabile scelta contiene la scelta utente riguardo le opzioni possibili

scelta:

```
.ascii "0000000000000000000"
```

Scelta è cosi definita in quanto quella è la capacità massima che il "buffer" è in grado di contenere, se il numero inserito avesse più di 19 caratteri andrebbe a compromettere le stampe successive.

A seconda del numero di caratteri di scelta e del valore effettivo viene selezionata una tra le opzioni edf, hpf o USCITA. Nel caso di USCITA viene terminata l'esecuzione del programma.

Intestazione.s

Funzione che si occupa di stampare a video il menù con le varie opzioni

---

Nome.s

Le funzioni di nome.s sono diverse e tra queste troviamo:

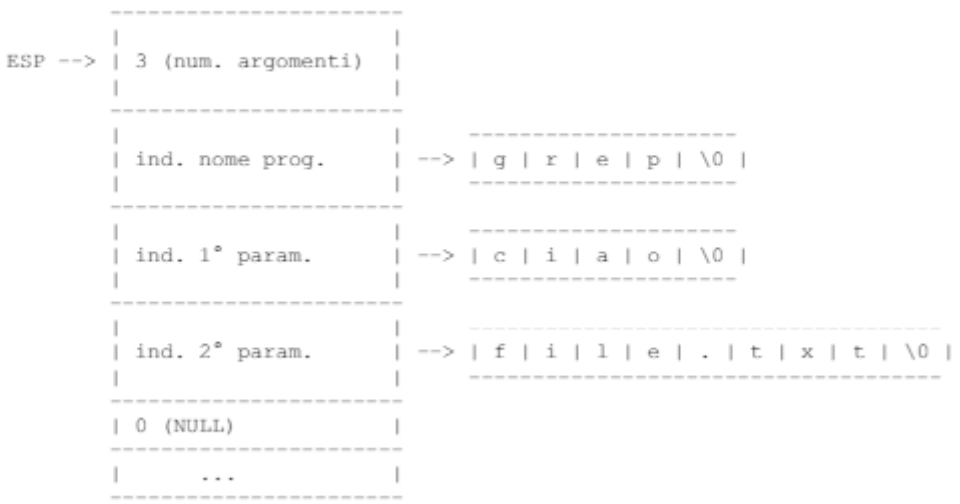
- prelevare l'indirizzo del file da cui acquisire i dati
- eventualmente anche quello su cui scrivere gli output
- trasformare i numeri acquisiti da formato stringa a intero
- controllare il range di tali numeri, inserirli nella pila (lavoriamo si questa per l'ordinamento)
- eventuali messaggi di errore sui file inseriti / dati acquisiti

Acquisizione degli indirizzi (file input &/ file output)

Il programma viene lanciato attraverso questo comando ./pianificatore fileInput eventualeFileOutput

La struttura che si presenta nella pila è molto simile a questa:

Ecco come si presenta lo stack appena viene avviato `grep ciao file.txt`



La cima della pila sarà quindi occupata dal numero di parametri letti, di conseguenza per capire se fossero stati inseriti i file su cui acquisire e su cui stampare è bastato scaricare il valore di esp in eax e fare dei check su quest'ultimo.

Eax == 1 -> legge un solo parametro che è l'eseguibile e quindi va in errore

Eax == 2 -> legge 2 parametri, l'eseguibile e il file da cui acquisire gli input

Eax == 3 -> legge 3 parametri, l'eseguibile, il file da cui acquisire gli input e il file per gli output

Eax > 3 -> legge troppi parametri e quindi va in errore

Successivamente abbiamo salvato l'indirizzo per la return (essendo una funzione viene codificato l'indirizzo dell'istruzione successiva e inserito nella pila, in questo modo la return sa dove tornare quando la funzione termina).

Acquisizione caratteri e trasformazione in interi

L'operazione di lettura consiste in una lettura in loop, carattere per carattere, del contenuto del file.

Se il carattere letto è == a "\n" (acapo), incremento la variabile lines che indica il numero di linee lette e quindi di prodotti acquisiti

Se il carattere letto è == a "," (virgola), lo trasformo in uno "\n", trasformando la linea prodotti in questo modo

`121,10,39,5\n -> 121\n10\n39\n5\n`

Se il carattere letto è diverso da a capo (\n) o dalla virgola (44), cerco l'intero corrispondente e salvo il numero che si sta formando nella variabile "numero". La formazione del numero termina quando viene letto un carattere "\n". Terminata la formazione del numero, pusho quest'ultimo nella pila.

Prima di pushare i numeri nella pila ci sono una serie di controlli sul range di questi valori:

```
1 <= id <= 127    1 <= durata <= 10      scadenza >= 1     1 <= priorità <= 5
```

Terminata l'acquisizione di valori, chiudo il file di acquisizione, carico in eax il numero di linee lette (necessarie per le operazioni di ordinamento) e carico in ebx ed ecx le variabili necessarie per la scrittura sul file di output.

Edf.s e Hpf.s

L'algorimo di ordinamento per i due file è di tipo selection sort (ricerco il valore con le caratteristiche minime e lo inserisco alla base della pila). L'algoritmo è stato prima sviluppato in C e successivamente "convertito" in linguaggio assembly con le dovute accortezze.

Funzionamento ordinamento:

Si presentano due cicli for. Il più esterno rappresenta il range di valori su cui lavorare, quello più interno permette di effettuare la selezione del valore minimo. Utilizziamo una variabile "check" per controllare che sia stato selezionato un valore minimo, questo per gestire l'eventualità che 2 prodotti abbiano i primi "campi" uguali e sia necessario controllare il secondo.

Alla fine del ciclo interno viene effettuata la sostituzione del prodotto minimo trovato con quello presente alla base (puntato dal ciclo for esterno).

I cicli continuano fino a quando non verranno ordinati tutti i prodotti presenti (una debolezza del selection sort è quella di non accorgersi se i prodotti inseriti sono già ordinati o meno).

L'algoritmo è molto simile per Hpf, sono solo cambiati i campi per i confronti effettuati.

Al termine dell'esecuzione di Edf o Hpf viene anche effettuata la stampa degli output su termine ed eventualmente su file.

La possibilità di stampare anche su file viene determinata dal valore della variabile "fileScrittura", una sorta di "check" che viene impostato a 1 in nomefile se durante il controllo degli input di esp viene individuata la presenta di 3 valori. Se il contatore lo permette, quindi, oltre a effettuare la stampa su terminale viene effettuata anche quella su file.

Il file viene creato se inesistente o viene modificato, se esistente, con l'ultimo output prodotto.

La stampa su file è molto simile a quella su terminare, cambia solo l'indirizzo di output (anzichéstdout viene utilizzato l'indirizzo del file di output).

PrintOutput.s

La stampa deve partire dai prodotti minori e arrivare a quelli maggiori. Di conseguenza è necessario spostarsi alla base della pila.

Stampiamo l'id del prodotto ->convertiamo l'intero corrispondente all'id in stringa

Stampiamo due punti ":"

Viene calcolato il tempo di inizio produzione del primo prodotto, per il primo prodotto l'inizio è 0 di conseguenza è un "caso particolare" che viene gestito separatamente dagli altri, nei successivi viene selezionato la durata del prodotto precedente e viene sommata al buffer già presente, in quanto rappresenterà il suo momento di inizio.

Questo procedimento di stampa è ciclico e viene effettuato per ogni prodotto ordinato inserito nella pila. Nel mentre di questo procedimento viene però anche calcolata la penalità, che rappresenta il ritardo del prodotto rispetto la sua scadenza. Il valore della penalità è quindi il ritardo * la priorità. Il calcolo viene effettuato per tutti i prodotti presenti e viene stampato al termine dopo il tempo di conclusione.

PrintOutputFile.s

La stampa su file è identica a quella su terminale, vengono effettuate le stesse operazioni, l'unica differenza vera e proprio è che anziché stampare su stdout (standard output), stampiamo sul file, quindi anziché avere ebx= 1 (per stdout), lo si collega all'indirizzo del file di output.