

Esercizi Assembly

Subroutine

Le subroutine sono frammenti di codice non autonomi utilizzati per rendere il codice più breve e leggibile. Permettono di creare delle funzioni all'interno di assembly.

Una subroutine è costruita così:

```
tag
STMFD r13!, {r5-r7, lr}    ; salvo i registri di lavoro e il link register
; CODICE...
LDMFD r13!, {r5-r7, pc}    ; ricarico i registri di lavoro
; carico il vecchio valore del link register sul program counter
```

Per chiamare una subroutine usiamo BL.

Esercizio 1

Scrivere un programma assembly che utilizza una subroutine per eseguire la divisione intera tra R0 e R1 restituendo quoziente e resto.

Esercizio 2

Scrivere un programma assembly che calcola la funzione di fibonacci restituendo $f(R0)$. Utilizzare una subroutine per implementare l'algoritmo.

Esercizio3

Scrivere un programma assembly per ordinare un vettore di interi salvato in memoria. Il programma riceve in input l'indirizzo di memoria del vettore e il numero di interi nel vettore rispettivamente in R0 e R1. L'algoritmo termina ritornando in R0 l'indirizzo di memoria del vettore ordinato.

Gestione della Memoria

Esercizio 1

Si consideri un sistema di scheduling per i processi: P_1, P_2, P_3, P_4, P_5 con le seguenti caratteristiche di tempi di esecuzioni ed ordine di priorità (ordinamento crescente, quindi

priorità 1 è la più alta mentre priorità 5 è la più bassa); si faccia riferimento alla tabella riportata qui di seguito. Dove nel caso di tempo di arrivo del caso A, i processi arrivano

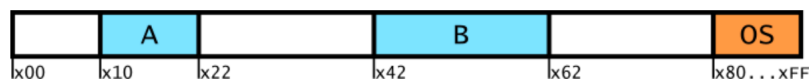
Processo N°	Tempo di Esecuzione	Priorità	Tempo di Arrivo (Caso A)	Tempo di Arrivo (Caso B)
P1	10	4	0	0
P2	1	5	0	1
P3	2	1	0	4
P4	1	3	0	6
P5	5	2	0	7

nell'ordine: P_1, P_2, P_3, P_4, P_5 .

- Si illustri [Caso A], l'ordine di esecuzione in caso si segua la politica SJF (Shortest Job First); si calcoli media del tempo di esecuzione e media del tempo di attesa.
- Si illustri [Caso A], l'ordine di esecuzione in caso si segua la politica LJF (Longest Job First); si calcoli media del tempo di esecuzione e media del tempo di attesa.

Esercizio 2

Un calcolatore che dispone di una memoria con ben 256 possibili indirizzi (da 0x00 a 0xFF) si trova, ad un certo punto, con la seguente configurazione in memoria:



- Quanti blocchi sono allocati rispettivamente per il processo A, il processo B ed il sistema operativo?
- Dei nuovi processi richiedono all'OS di essere allocati in memoria (in questo ordine):
 1. Il processo C richiede 10 blocchi
 2. Il processo D richiede 20 blocchi
 3. Il processo E richiede 10 blocchi

Illustrare con un diagramma simile a quello presente nel testo dell'esercizio (quindi specificando esplicitamente gli indirizzi) la configurazione di memoria del calcolatore a seguito dell'allocazione dei processi specificati nei casi in cui la memoria venga gestita secondo i diversi paradigmi *first-fit*, *next-fit*, *best-fit* e *worst-fit* (Nota: l'ultimo processo a essere stato allocato prima di questi tre è il processo B).