# Prova pratica di Calcolatori Elettronici (nucleo v6.*)

*C.d.L. in Ingegneria Informatica, Ordinamento DM 270*

6 luglio 2011

1. Vogliamo aggiungere al nucleo il meccanismo delle interruzioni inter-processo. Un qualunque processo può inviare una interruzione ad un altro processo di cui conosce l'identificatore. L'interruzione setta un flag nel descrittore del processo destinatario. Ogni processo può esaminare lo stato del proprio flag con una opportuna primitiva.

   Se il processo destinatario si era sospeso sulla coda del timer con la primitiva `delay(natl n)`, l'operazione di interruzione, invece di settare il flag, risveglia il processo. La primitiva `delay(natl n)` può ora terminare per due motivi distinti: o perché sono trascorsi `n` intervalli di tempo, o perché il processo è stato interrotto. Per poter distingure questi due casi la primitiva restituisce un valore al chiamante. Tale valore è il numero di intervalli di tempo che il processo avrebbe dovuto ancora attendere (e dunque è 0 se il processo si è risvegliato normalmente).

   A tale scopo aggiungiamo i seguenti campi al descrittore di ogni processo:

```
bool  interrupted;
bool  sleeping;
```

   Il campo `interrupted` è il flag di interruzione, posto a `false` alla creazione del processo. Il campo `sleeping` è un flag che vale `true` se il processo si trova nella coda del timer, e `false` altrimenti.

   Modifichiamo inoltre la primitiva `delay` in modo che restituisca un `natl`, e le funzioni `c_delay` e `c_driver_td` in modo che gestiscano il flag `sleeping` e il valore di ritorno della `delay` (per le modifiche a queste due funzioni si rimanda al file `sistema.cpp` fornito)

   Aggiungiamo infine le seguenti primitive:

   - `bool interrupt(natl id)`: Se `id` non corrisponde ad un processo esistente restituisce `false` e termina. In tutti gli altri casi restituisce `true`. Se il flag `interrupted` è già a `true` non fa altro. Se il flag `interrupted` è `false` e il processo non è nella coda del timer, si limita a settare l'opportuno flag. Infine, se il flag `interrupted` è false ed il processo si trova nella coda del timer lo risveglia. Per risvegliarlo deve rimuoverlo dalla coda del timer, avendo cura di riportarla in uno stato consistente. Aggiorna opportunamente i campi del descrittore di processo e gestisce eventuali *preemption*.
   - `bool interrupted()` (già realizzata): restituisce il valore del flag `interrupt` del processo corrente e lo resetta.

   Modificare i file `sistema.cpp` e `sistema.s` in modo da realizzare la primitiva `interrupt()`.