Prova pratica di Calcolatori Elettronici (nucleo v6.*)

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

29 luglio 2020

1. Vogliamo permettere ad ogni processo di definire ciò che deve succedere quanto esso stesso causa una eccezione che, normalmente, ne causarebbe la terminazione forzata. Forniamo dunque una primitiva `signal(tipo, sighandler)`, tramite la quale un processo può associare un proprio *gestore* `sighandler` alle eccezioni di tipo `tipo`. Da quel momento in poi, quando il processo in questione causa una eccezione di questo tipo, il nucleo fa in modo che il processo stesso passi ad eseguire il gestore (ovviamente, al livello di privilegio del processo). Il gestore può poi terminare o abortire il processo (invocando le normali privitive già esistenti), oppure può decidere di farne proseguire l'esecuzione invocando la primitiva `signal_return(rip)`, che termina il gestore e fa proseguire il processo dall'indirizzo `rip`.

   Ogni processo può associare un gestore, anche diverso, ad ogni eccezione (tranne la numero 2, associata agli interrupt non mascherabili). Le eccezioni che non sono associate a gestori si comportano come sempre, causando la terminazione forzata del processo che le ha causate. L'eccezione di tipo page-fault deve essere sempre gestita normalmente e passare il controllo al gestore solo nel caso in cui la normale gestione ha causato un errore.

   I gestori hanno il seguente tipo:

   ```
   typedef void (*sighandler)(int tipo, natq errore, natq rip)
   ```

   I gestori devono dunque ricevere tre parametri: il `tipo` dell'eccezione (utile nel caso in cui lo stesso gestore sia stato associato a più eccezioni), l'`errore` che specifica ulteriormente la causa dell'eccezione, e il `rip` salvato in pila dal meccanismo delle eccezioni. Nel caso dell'eccezione di page-fault, `errore` contiene l'indirizzo virtuale che ha causato il fault.

   I gestori non devono a loro volta causare eccezioni: se ciò accade, il processo deve essere abortito.

   Aggiungiamo le seguenti primitive (abortiscono il processo in caso di errore):

   - `void signal(int tipo, sighandler sigh)`: associa il signal handler `sigh` all'eccezione di tipo `tipo`, sostituendo un eventuale signal handler precedente. Se `sigh` vale zero viene ripristinato il funzionamento normale dell'eccezione. È un errore se `tipo` non corrisponde a nessun tipo di eccezione, oppure se si tenta di associare il gestore all'eccezione numero 2.
   - `void signal_return(natq rip)`: termina il gestore e fa ripartire il processo dall'istruzione `rip`. I registri generali del processore devono avere il contenuto che avevano al momento dell'eccezione. È un errore invocare questa primitiva se non è in corso la gestione di una eccezione.

   Modificare i file `sistema.s` e `sistema.cpp` in modo da realizzare il meccanismo descritto.