Prova pratica di Calcolatori Elettronici (nucleo v6.*)

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

17 settembre 2014

1. Una *barriera* è un meccanismo di sincronizzazione tra processi che funziona nel modo seguente: la barriera è normalmente chiusa; se un processo arriva alla barriera si blocca; la barriera si apre solo quando sono arrivati tutti i processi registrati su quella barriera. Quando la barriera si apre tutti i processi si sbloccano, la barriera si richiude (istantaneamente) e il meccanismo si ripete.

   Vogliamo realizzare il meccanismo della barriera nel nucleo, prevedendo la possibilità che esistano più barriere distinte, identificate con un numero da 0 a $MAX\_BARRIERS - 1$. In ogni istante, ogni processo è registrato su al più una barriera e può sincronizzarsi solo sulla barriera su cui è eventualmente registrato. La registrazione di un processo su una barriera può essere richiesta da un qualunque processo, non necessariamente coincidente con il processo da registrare, e può essere cambiata in un qualunque momento, anche quando il processo da registrare è già bloccato su un'altra barriera. In quest'ultimo caso il processo viene spostato sulla nuova barriera, come se si fosse sincronizzato su quest'ultima invece che sulla precedente.

   Le barriere sono create all'avvio del sistema e non possono essere distrutte.

   Per rappresentare una barriera introduciamo le seguenti strutture dati:

   ```
   struct barrier {
       natl nproc;
       natl narrived;
       des_proc *waiting;
   };

   barrier barriers[MAX_BARRIERS];
   ```

   Dove: `nproc` è il numero di processi registrati sulla barriera; `narrived` conta i processi arrivati alla barriera dall'ultima apertura (o dalla creazione, quando la barriera non è ancora mai stata aperta); `waiting` è la coda dei processi che attendono l'apertura della barriera. L'array `barriers` contiene tutte le barriere del sistema. L'identificatore di una barriera è il suo indice all'interno dell'array.

   Aggiungiamo inoltre i seguenti campi al descrittore di processo:

   ```
   natl barrier_id;
   bool waiting;
   ```

   Dove: `barrier_id` è l'identificatore della barriera su cui il processo è registato (`0xFFFFFFFF` se nessuna); `waiting` è `true` se e solo se il processo è bloccato sulla barriera.

   Aggiungiamo infine le seguenti primitive:

   - `void barrier()` (da realizzare): fa giungere il processo corrente alla barriera su cui è registrato. È un errore se il processo non è registrato su alcuna barriera.

- `bool reg(natl pid, natl bid)` (da realizzare): registra il processo di identificatore `pid` sulla barriera di identificatore `bid`, o su nessuna barriera se `bid` è `0xFFFFFFFF`. È un errore tentare di registrare il processo su una barriera che non esiste. La primitiva restituisce `false` se il processo non esiste, e `true` in tutti gli altri casi.

Le primitive abortiscono il processo chiamante in caso di errore e tengono conto della priorità tra i processi.

Modificare i file `sistema.cpp` e `sistema.s` in modo da realizzare le primitive mancanti.