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

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

16 gennaio 2020

1. Vogliamo fornire ai processi la possibilità di bloccarsi in attesa che un altro processo riceva una eccezione o termini. Un processo $P$ deve prima registrarsi, tramite la primitiva `proc_attach(natl id)`, con il processo di identificatore `id`, chiamiamolo $Q$, di cui vuole controllare la terminazione. Diremo che $P$ è il *master* di $Q$ e che $Q$ è lo *slave* di $P$. Successivamente il processo $P$ può invocare la primitiva `proc_wait()` per bloccarsi in attesa che il processo $Q$ termini (invocando `terminate_p()`) o riceva una eccezione. La primitiva `proc_wait()` restituisce al processo $P$ il numero dell'eccezione ricevuta da $Q$, o il valore 32 in caso di terminazione normale. Si noti che la gestione dell'eccezione da parte del processo $Q$ non cambia anche con questo nuovo meccanismo (quindi il processo $Q$ deve essere comunque abortito, si veda `gestore_eccezioni()` in `sistema/sistema.cpp`).

   Si modifichino i file `sistema/sistema.s` e `sistema/sistema.cpp` per implementare le seguenti primitive (abortiscono il processo in caso di errore):

   - `bool proc_attach(natl id)`: (tipo 0x59, da realizzare) La primitiva restituisce `false` se il processo che la invoca è uno slave, oppure se il processo `id` non esiste oppure è già un master o uno slave. È un errore se il processo $P$ è già master o cerca di diventare master di se stesso. Altrimenti fa in modo che $P$ diventi il master di `id` e restituisce `true`.
   - `natl proc_wait()`: (tipo 0x5a, da realizzare): attende che il processo slave termini, normalmente o per la ricezione di una eccezione (nota: si trascurino i page faulti, tipo 14, e le interruzioni non mascherabili, tipo 2) e restituisce il numero dell'eccezione, o 32 nel caso di terminazione normale. È un errore invocare questa primitiva se il processo non è master;