Prova pratica di Calcolatori Elettronici (nucleo v6.*)

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

12 giugno 2019

1. Vogliamo fornire ai processi la possibilità di scoprire se l'esecuzione di un altro processo passa da una certa istruzione. Per far questo forniamo una primitiva `breakpoint(vaddr rip)` che installa un breakpoint (istruzione `int3`, codice operativo `0xCC`) all'indirizzo `rip`, quindi blocca il processo chiamante, sia P_1. Quando (e se) un altro processo P_2 arriva a quell'indirizzo, il processo P_1 deve essere risvegliato. Si noti che il processo P_2 non si blocca e deve proseguire la sua esecuzione indisturbato (salvo che potrebbe dover cedere il processore a P_1 per via della preemption).

   Prevediamo le seguenti limitazioni del meccanismo:

   1. per ogni chiamata di `breakpoint(rip)` viene intercettato solo il primo processo che passa da `rip`: altri processi che dovessero passarvi dopo il primo non vengono intercettati;
   2. Un solo processo alla volta può chiamare `breakpoint()`; la primitiva restituisce un errore se un altro processo sta già aspettando un breakpoint.

   Si noti che se un processo esegue `int3` senza che ciò sia richiesto da una primitiva `breakpoint()` attiva, il processo deve essere abortito.

   Si modifichino dunque i file `sistema/sistema.s` e `sistema/sistema.cpp` per implementare la seguente primitiva:

   - `natl breakpoint(vaddr rip)`: (tipo 0x59): blocca il processo chiamante in attesa che un altro processo provi ad eseguire l'istruzione all'indirizzo `rip`; restituisce l'id del processo intercettato, o 0xFFFFFFFF se un altro processo sta già aspettando un breakpoint (a qualunque indirizzo); abortisce il processo se `rip` non appartiene all'intervallo [`ini_utn_c`, `fin_utn_c`) (zona utente/condivisa).