Prova pratica di Calcolatori Elettronici

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

23 settembre 2019

1. Siano date le seguenti dichiarazioni, contenute nel file cc.h:

```
struct st1 { char vc[4]; }; struct st2 { char vd[4]; };
class cl {
        long v[4];
        st1 c1; st1 c2;
public:
        cl(char c, st2& s);
        void elab1(st1 s1, st2 s2);
        void stampa()
        {
            for (int i=0; i < 4; i++) cout << v[i] << ' '; cout << "\n";
            for (int i=0; i < 4; i++) cout << c1.vc[i] << ' '; cout << "\n";
            for (int i=0; i < 4; i++) cout << c2.vc[i] << ' '; cout << "\n\n";
        }
};</pre>
```

Realizzare in Assembler GCC le funzioni membro seguenti.

```
cl::cl(char c, st2& s2) {
    for (int i = 0; i < 4; i++) {
        c1.vc[i] = c; c2.vc[i] = c++;
        v[i] = s2.vd[i] + c2.vc[i];
    }
}
void cl::elab1(st1 s1, st2 s2) {
    cl cla('a', s2);
    for (int i = 0; i < 4; i++) {
        if (c2.vc[i] <= s1.vc[i]) {
            c1.vc[i] = i + cla.c2.vc[i];
            v[i] = i - cla.v[i];
        }
}</pre>
```

2. Vogliamo fornire ai processi la possibilità di bloccare l'esecuzione di un processo a scelta, ogni volta che questo passa da una certa istruzione. Per far questo forniamo alcune primitive. Con la primitiva bpattach(natl id, vaddr rip) un processo master installa un breakpoint (istruzione int3, codice operativo 0xCC) all'indirizzo rip nella memoria virtuale del processo id, che diventa slave. Da quel momento in poi il processo slave si blocca se passa da rip. Nel frattempo, usando la primitiva bpwait(),

il processo master può sospendersi in attesa che lo slave passi dal breakpoint. A quel punto il processo master può invocare la primitiva **bpcontinue()** per permettere al processo slave di continuare la propria esecuzione come se non fosse mai stato intercettato. Se lo slave ripassa dal breakpoint viene intercettato nuovamente e il meccanismo di ripete. Infine, con la primitiva **bpdetach()**, il processo master rimuove il breakpoint e, se necessario, risveglia un'ultima volta lo slave.

Si noti che processi che non sono slave non devono essere intercettati. Inoltre, se un processo esegue int3 senza che ciò sia stato richiesto da un master, il processo deve essere abortito.

Aggiungiamo i seguenti campi ai descrittori di processo:

```
des_proc *bp_master;
des_proc *bp_slave;
vaddr bp_addr;
natb bp_orig;
natl bp_slave_id;
struct proc_elem *bp_waiting;
```

dove: bp_master punta al processo master di questo processo (nullo se il processo non ha un master); bp_slave punta al processo slave di questo processo (nullo se il processo non ha uno slave); bp_addr e bp_orig sono sinificativi solo per i processi slave e contengono, rispettivamente, l'indirizzo a cui è installato il breakpoint e il byte originariamente contenuto a quell'indirizzo; il campo bp_slave_id è significativo solo per il processo master e contiene l'id dello slave; bp_waiting è una coda su cui i processi master e slave si possono bloccare: il master su quella dello slave e viceversa.

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

- bool bpattach(natl id, vaddr rip): (tipo 0x59, già realizzata): se il processo che invoca la primitiva non è uno slave e il processo id esiste e non è già uno slave o un master, installa il breakpoint all'indirizzo rip e restituisce true, altrimenti restituisce false; è un errore se rip non appartiene all'intervallo [ini_utn_c, fin_utn_c) (zona utente/condivisa) o se il processo è già master o cerca di diventare master di se stesso.
- void bpwait(): (tipo 0x5a, già realizzata): attende che il processo slave passi dal breakpoint; è un errore invocare questa primitiva se il processo non è master;
- void bpcontinue(): (tipo 0x5c, da realizzare): permette allo slave di proseguire l'esecuzione, facendo in modo che venga intercettato nuovamente se ripassa dal breakpoint; è un errore invocare questa primitiva se il processo non è master o se lo slave non è bloccato sul breakpoint;
- void bpdetach() (tipo 0x5b, già realizzata): disfa tutto ciò che ha fatto la bpattach() e risveglia eventualmente il processo slave; è un errore invocare questa primitiva se il processo non è master;

Suggerimento: per realizzare la bpcontinue() si deve temporaneamente rimuovere il breakpoint, quindi reinserirlo non appena lo slave ha eseguito una istruzione. Per intercettare questo evento si può abilitare temporaneamente il single-step trap nel processo slave.