Prova pratica di Calcolatori Elettronici

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

12 giugno 2019

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

```
struct st1 {
        char vc[4];
};
class cl {
        long v[4];
        st1 s;
public:
        cl(char c, st1 s2);
        void elab1(st1& s1);
        void stampa()
                 for (int i = 0; i < 4; i++) cout << s.vc[i] << ', '; cout << endl;
                 for (int i = 0; i < 4; i++) cout << v[i] << ', '; cout <math><< endl << endl;
        }
};
Realizzare in Assembler GCC le funzioni membro seguenti.
cl::cl(char c, st1 s2)
{
        for (int i = 0; i < 4; i++) {
                s.vc[i] = c;
                v[i] = s2.vc[i] - c;
        }
}
void cl::elab1(st1& s1)
        cl cla('x', s1);
        for (int i = 0; i < 4; i++) {
                if (s.vc[i] <= s1.vc[i]) {
                         s.vc[i] = cla.s.vc[i];
                         v[i] = cla.v[i] + i;
                }
        }
}
```

2. 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).

Suggerimento: Il comando process dump del debugger è stato modificato in modo da mostrare il disassemblato del codice intorno al valore di rip salvato in pila.