Prova pratica di Calcolatori Elettronici

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

28 giugno 2023

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

```
struct st {
        char vv1[4];
        long vv2[4];
};
class cl {
        st s;
public:
        cl(char v[]);
        void elab1(int d, st& ss);
        void stampa()
        {
                for (int i = 0; i < 4; i++)
                        cout << (int)s.vv1[i] << ' ';
                cout << '\t';
                for (int i = 0; i < 4; i++)
                        cout << s.vv2[i] << ' ';
                cout << endl;
                cout << endl;
        }
};
```

   Realizzare in Assembler GCC le funzioni membro seguenti.

```
void cl::elab1(int d, st& ss)
{
        for (int i = 0; i < 4; i++) {
                if (d <= ss.vv2[i])
                        s.vv1[i] -= ss.vv1[i];
                s.vv2[i] = d + i;
        }
}
```

2. Aggiungiamo al sistema la primitiva

```
bool sem_wait_to(natl sem, natl to);
```

   che funziona come la `sem_wait()`, ma in più permette di specificare un *time-out* `to` per il caso in cui il semaforo `sem` non contenga gettoni. Più in dettaglio: se il semaforo `sem` contiene almeno un gettone, il processo lo prende senza bloccarsi; altrimenti si sospende e a quel punto possono succedere due cose:

1. prima che `to` sia trascorso, qualche processo esegue una `sem_signal()` che risveglia il processo; in quel caso il processo prende il gettone e `sem_wait_to()` restituisce `true`;
2. trascorre il tempo `to` senza che nessuno risvegli il processo tramite una `sem_signal()`; in questo caso il processo si risveglia (senza aver preso un gettone da `sem`) e `sem_wait_to()` restituisce `false`.

È ammesso il caso `to==0`, e in quel caso la `sem_wait_to()` si comporta nel seguente modo: se il semaforo contiene un gettone lo prende e restituisce `true`, altrimenti non fa niente e restituisce `false`.

Per realizzare la nuova primitiva agiungiamo un campo `natl waiting_on` ai descrittori dei processi. Il campo contiene un valore diverso da `0xFFFFFFFF` solo se il processo è sospeso in attesa che si verifichi uno dei due eventi elencati sopra. La primitiva `sem_wait_to()`, se necessario, inserisce il processo sia nella coda del semaforo, sia nella coda del timer (`p_sospesi`). Modifichiamo poi la funzione `c_sem_signal()` in modo che gestisca il caso 1, e la funzione `c_driver_td()` in modo che gestisca il caso 2.

Modificare il file `sistema.cpp` in modo da completare le parti mancanti.