Prova pratica di Calcolatori Elettronici

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

18 gennaio 2017

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

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

   Realizzare in Assembler GCC le funzioni membro seguenti.

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

2. Introduciamo un meccanismo di *broadcast* tramite il quale un processo può inviare un messagio ad un insieme di processi. Per ricevere o inviare un broadcast i processi si devono preventivamente registrare come *listener* o *broadcaster*, rispettivamente. Un solo processo alla volta può essere registrato come broadcaster (un nuovo processo può diventare broadcaster solo quando il precedente termina).

Il sistema ricorda tutti i messaggi di broadcast inviati (fino ad un massimo dato dalla costante `MAX_BROADCAST`) e ciascun processo listener li riceve tutti, in ordine. I messaggi sono di tipo `natl`.

Per realizzare il sistema aggiungiamo il seguente tipo enumerato:

```
enum broadcast_role { B_NONE, B_BROADCASTER, B_LISTENER };
```

e il seguente campo ai descrittori di processo:

```
broadcast_role b_reg;
```

Il campo è posto a `B_NONE` alla creazione del processo.

Aggiungiamo infine le seguenti primitive:

- `void reg(broadcast_role role)` (tipo 0x3a, da realizzare): registra il processo per il ruolo dato da `role`; è un errore se role non specifica né un broadcaster, né un listener, se il processo era già registrato (per lo stesso o un altro ruolo), o se cè già un broadcaster e si tenta di registrarne un altro;
- `natl listen()` (tipo 0x3b, da realizzare): restituisce il prossimo messagio di broadcast non ancora letto dal processo; se il processo li ha già letti tutti, si blocca in attesa del prossimo; è un errore se il processo non è registrato come listener;
- `void broadcast(natl msg)` (tipo 0x3c, da realizzare): invia in broadcast il messaggio `msg`; è un errore se il processo non è registrato come broadcaster o se il limite di messaggi di broadcast è stato superato.

Le primitive abortiscono il processo chiamante in caso di errore e tengono conto della priorità tra i processi.

Modificare i file `sistema.cpp` e `sistema.S` in modo da realizzare le primitive mancanti. Attenzione: il candidato deve definire anche le necessarie strutture dati.