# Prova pratica di Calcolatori Elettronici

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

25 luglio 2018

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

```
struct st {
        long vv2[4];
        char vv1[4];
};
class cl {
        st s;
public:
        cl(char v[]);
        void elab1(st& ss, int d);
        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.

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

2. Colleghiamo al sistema una perifierica PCI di tipo `ce`, con vendorID `0xedce` e deviceID `0x1234`. Le periferiche `ce` sono schede di rete che operano in PCI Bus Mastering. Il software deve preparare dei buffer vuoti, che la scheda riempie autonomamente con messaggi ricevuti dalla rete.

   La scheda usa una coda circolare di descrittori di buffer. Ogni descrittore deve contenere l'indirizzo fisico di un buffer e la sua lunghezza in byte. La coda di buffer ha 8 posizioni, numerate da 0 a 7.

   La scheda possiede due registri, `HEAD`, di sola lettura, e `TAIL`, di lettura/scrittura. I due registri contengono numeri di posizioni e inizialmente sono entrambi pari a zero. La scheda può usare soltanto i descrittori che vanno da `HEAD` in avanti (circolarmente) senza toccare `TAIL`. Inizialmente, dunque, la scheda non può usare alcun descrittore. Il software deve allocare dei buffer a partire dal descrittore puntato da `TAIL` e poi scrivere in `TAIL` l'indice del primo descrittore che la scheda non può usare. Conviene allocare sempre il massimo numero possibile di buffer, perché la scheda butta via i messaggi ricevuti quando non ha a disposizione buffer in cui copiarli. Si noti che il massimo numero di desrittori che la scheda può usare è pari a 7 (dimensione della coda meno 1), in quanto la configurazione con `HEAD` uguale a `TAIL` è interpretata dalla scheda come "coda vuota".

   Ogni volta che la scheda ha terminato di ricevere un messaggio incrementa (modulo 8) il contenuto di `HEAD` e, se non sta aspettando una risposta ad una richiesta di interruzione precedente, invia una nuova richiesta di interruzione. La lettura di `HEAD` funge da risposta alla richiesta. È dunque possibile (e, anzi, normale) che quando il software legge `HEAD` questo sia avanzato di più di una posizione rispetto all'ultima lettura: vuol semplicemente dire che tra le due letture la scheda ha finito di ricevere più di un messaggio. I descrittori dei messaggi ricevuti saranno quelli che si trovano tra l'ultima posizione letta da `HEAD` (inclusa) e la nuova (esclusa).

   Ogni periferica `ce` usa 16 byte nello spazio di I/O a partire dall'indirizzo base specificato nel registro di configurazione BAR0, sia $b$. I registri accessibili al programmatore sono i seguenti:

   1. **HEAD** (indirizzo $b$, 4 byte): posizione di testa;
   2. **TAIL** (indirizzo $b + 4$, 4 byte): posizione di coda;
   3. **RING** (indirizzo $b + 8$, 4 byte): indirizzo fisico del primo descrittore della coda circolare;

   Supponiamo che ogni computer collegato alla rete possieda un indirizzo numerico di 4 byte. Ogni computer possiede un'unica perifica di tipo `ce`. I messaggi che viaggiano sulla rete contengono una "intestazione" con quattro campi (ciascuno grande 4 byte): l'indirizzo del computer che invia, l'indirizzo `dst` del computer a cui il messaggio è destinato, un numero di *porta* (compreso tra 0 e 15) e la lunghezza `len` del resto del messaggio (esclusa l'intestazione). L'intestazione è seguita dal messaggio vero e proprio, di lunghezza massima `MAX_PAYLOAD`.

   Vogliamo fornire all'utente una primitiva

   ```
   bool receive(natl port, char *msg, natq& len)
   ```

   che permetta di ricevere nel buffer `msg` un messaggio destinato alla porta `port`. Il parametro `len` contiene inizialmente la dimensione del buffer e, dopo la ricezione, contiene la dimensione effettiva del messaggio ricevuto. Attenzione: l'utente deve ricevere in `msg` solo il messaggio vero e proprio, esclusa l'intestazione. Si noti che ogni invocazione della primitiva restituisce un solo messaggio: eventuali altri messaggi in coda verranno restituiti alla prossime invocazioni. Se, invece, non vi sono messaggi in coda, la primitiva blocca il processo in attesa che ne arrivi almeno uno. La primitiva restituisce `false` se il buffer `msg` non è sufficiente a contenere il prossimo messaggio da ricevere o se `port` non è valido, e `true` altrimenti.

   Si può assumere che la scheda riceva solo messaggi effettivamente destinati al computer a cui è collegata.

   Modificare i file `io.S` e `io.cpp` completando le parti mancanti.

   **NOTA:** le strutture dati necessarie sono tutte descritte nel file `io.cpp`, all'interno dei marcatori `ESAME`.