

# Prova pratica di Calcolatori Elettronici

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

6 luglio 2016

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

```
struct st1 { char vi[4]; }; struct st2 { int vd[4]; };
class cl
{
    char v1[4]; char v2[4]; long v3[4];
public:
    cl(st1 ss); cl(st1 s1, long ar2[]);
    cl elab1(char ar1[], st2 s2);
    void stampa()
    {
        char i;
        for (i=0;i<4;i++) cout << (int)v1[i] << ' '; cout << endl;
        for (i=0;i<4;i++) cout << (int)v2[i] << ' '; cout << endl;
        for (i=0;i<4;i++) cout << v3[i] << ' '; cout << endl << endl;
    }
};
```

Realizzare in Assembler GCC le funzioni membro seguenti.

```
cl::cl(st1 ss)
{
    for (int i = 0; i < 4; i++) {
        v1[i] = v2[i] = ss.vi[i]; v3[i] = ss.vi[i] + ss.vi[i];
    }
}
cl::cl(st1 s1, long ar2[])
{
    for (int i=0; i<4; i++) {
        v1[i] = v2[i] = s1.vi[i]; v3[i] = ar2[i];
    }
}
cl cl::elab1(char ar1[], st2 s2)
{
    st1 s1;
    for (int i = 0; i < 4; i++)
        s1.vi[i] = ar1[i];
    cl cla(s1);
    for (int i = 0; i < 4; i++)
        cla.v3[i] = s2.vd[i];
    return cla;
}
```

2. Colleghiamo al sistema delle periferiche PCI di tipo *ce*, con vendorID `0xedce` e deviceID `0x1234`. Ogni periferica *ce* usa 16 byte nello spazio di I/O a partire dall'indirizzo base specificato nel registro di configurazione BAR0, sia *b*.

Le periferiche *ce* sono periferiche di ingresso in grado di generare interruzioni. I registri accessibili al programmatore sono i seguenti:

1. **CTL** (indirizzo *b*, 1 byte): registro di controllo; il bit numero 0 permette di abilitare (1) o disabilitare (0) le richieste di interruzione;
2. **STS** (indirizzo *b* + 4, 1 byte): registro di stato; il bit numero 0 vale 1 se e solo se il registro RBR contiene un dato non ancora letto;
3. **RBR** (indirizzo *b* + 8, 1 byte): registro di lettura;

L'interfaccia genera una interruzione se le interruzioni sono abilitate e il registro RBR contiene un valore non ancora letto. L'interfaccia non presenta nuovi valori in RBR se questo ne contiene uno non ancora letto, quindi la lettura di RBR funge da risposta alla richiesta di interruzione.

Vogliamo fornire all'utente una primitiva

```
ceread(natl id, char *buf, natl& quanti, char stop)
```

Il parametro *id* identifica una delle periferiche *ce* installate. La primitiva permette di leggere da tale periferica una sequenza di byte che termina con il carattere *stop* passato come quarto argomento. I byte letti saranno scritti a partire dall'indirizzo *buf*. Il parametro *quanti* è usato sia come argomento di ingresso che di uscita: in ingresso l'utente specifica il numero massimo di byte da leggere (anche se *stop* non è stato ricevuto) e in uscita la primitiva dice all'utente il numero di byte che sono stati effettivamente letti (che può essere inferiore al massimo, quando si riceve *stop*).

Per descrivere le periferiche *ce* aggiungiamo le seguenti strutture dati al modulo I/O:

```
des_ce array_ce[MAX_CE];  
natl next_ce;
```

I primi *next\_ce* elementi del vettore *array\_ce* contengono i destrittori, opportunamente inizializzati, delle periferiche di tipo *ce* effettivamente rilevate in fase di avvio del sistema. Ogni periferica è identificata dall'indice del suo descrittore. La struttura *des\_ce* deve essere definita dal candidato.

Modificare i file *io.s* e *io.cpp* in modo da realizzare la primitiva come descritto.