

Prova pratica di Calcolatori Elettronici

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

24 febbraio 2017

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

```
struct st1 { int vi[4]; };
struct st2 { char vd[4]; };
class cl {
    char v1[4]; char v3[4]; long v2[4];
public:
    cl(st1 ss);
    cl(st1& s1, int ar2[]);
    cl elab1(char ar1[], st2 s2);
    void stampa() {
        for (int i = 0; i < 4; i++) cout << (int)v1[i] << ' '; cout << endl;
        for (int i = 0; i < 4; i++) cout << (int)v2[i] << ' '; cout << endl;
        for (int i = 0; i < 4; i++) cout << (int)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] = ss.vi[i]; v2[i] = ss.vi[i] / 2;
        v3[i] = 2 * ss.vi[i];
    }
}
cl::cl(st1& s1, int ar2[])
{
    for (int i = 0; i < 4; i++) {
        v1[i] = s1.vi[i]; v2[i] = s1.vi[i] / 4;
        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] + 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 32 byte nello spazio di I/O a partire dall'indirizzo base specificato nel registro di configurazione BAR0, sia **b**.

La periferiche **ce** sono periferiche di ingresso in grado di operare in PCI Bus Mastering. Sono inoltre in grado di eseguire tutte le necessarie trasformazioni da indirizzi virtuali a fisici, utilizzando le stesse strutture dati della MMU del processore, purchè non incontrino bit P a zero durante la traduzione.

I registri accessibili al programmatore, tutti di 4 byte, sono i seguenti:

1. **VPTRHI** (indirizzo **b**): parte più significativa dell'indirizzo virtuale di destinazione
2. **VPTRLO** (indirizzo **b + 4**): parte meno significativa dell'indirizzo virtuale di destinazione;
3. **CNT** (indirizzo **b + 8**): numero di byte da trasferire;
4. **CR3** (indirizzo **b + 12**): indirizzo fisico della tabella di livello 4;
5. **STS** (indirizzo **b + 16**): registro di stato;
6. **CMD** (indirizzo **b + 20**): registro di comando.

Ogni volta che si scrive il valore 1 nel registro **CMD**, la periferica tenta di scrivere **CNT** byte in memoria a partire dall'indirizzo virtuale contenuto in **VPTRHI**, **VPTRLO**. Gli indirizzi verranno tradotti utilizzando la tabella di corrispondenza puntata dal registro **CR3** dell'interfaccia. Se l'interfaccia incontra degli errori durante la traduzione (per es. un bit P a zero), interrompe il trasferimento e setta il bit 2 di **STS**. In ogni caso la periferica invia una richiesta di interruzione al completamento dell'operazione (o perché non ha più byte da trasferire, o perché ha riscontrato un errore).

Le interruzioni sono sempre abilitate. La lettura del registro di stato funziona da risposta alle richieste di interruzione.

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

```
bool cedmaread(natl id, natl quanti, char *buf)
```

che permette di leggere **quanti** byte dalla periferica numero **id** (tra quelle di questo tipo), copiandoli nel buffer **buf**. La primitiva restituisce **false** se il trasferimento è stato interrotto per errori, **true** altrimenti.

Controllare tutti i problemi di Cavallo di Troia.

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

```
struct des_ce {
    ioaddr iVPTRHI, iVPTRLO, iCNT, iCR3, iSTS, iCmd;
    natl sync;
    natl mutex;
    bool error;
};
des_ce array_ce[MAX_CE];
natl next_ce;
```

La struttura **des_ce** descrive una periferica di tipo **ce** e contiene al suo interno gli indirizzi dei vari registri, l'indice di un semaforo inizializzato a zero (**sync**), l'indice di un semaforo inizializzato a 1 (**mutex**) e un booleano per memorizzare il successo o fallimento dell'ultima operazione.

I primi **next_ce** elementi del vettore **array_ce** contengono i descrittori, opportunamente inizializzati, delle periferiche di tipo **ce** effettivamente rilevate in fase di avvio del sistema. Ogni periferica è identificata dall'indice del suo descrittore.

Nota: il modulo sistema mette a disposizione del modulo di I/O le primitive

```
natl resident(void addr, natl quanti);  
void nonresident(natl id);
```

La primitiva `resident()` permette di caricare in memoria e rendere temporaneamente non rimpiazzabili tutte le pagine virtuali (e le relative tabelle) che contengono gli indirizzi da `addr` a `addr+quanti` (escluso). La primitiva restituisce un identificatore di operazione che può essere successivamente passato a `nonresident()` per rendere nuovamente rimpiazzabili le pagine (e le tabelle) in questione. In caso di errore restituisce `0xffffffff`.