

Prova pratica di Calcolatori Elettronici

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

27 luglio 2016

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

```
struct st {
    char vv1[4];
    long vv2[4];
};
class cl {
    char a, b;
    st s;
public:
    cl();
    cl(char v[]);
    void elab1(st& ss, int d);
    void stampa()
    {
        cout << (int)a << ' ' << (int)b << endl;
        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() { }
cl::cl(char v[])
{
    a = v[0]++;
    b = v[1];
    for (int i = 0; i < 4; i++) {
        s.vv1[i] = v[i] + a;
        s.vv2[i] = v[i] + b;
    }
}
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] = a + d;
    }
}

```

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**.

La periferiche **ce** sono periferiche di ingresso in grado di operare in PCI Bus Mastering. I registri accessibili al programmatore sono i seguenti:

1. **BMPTR** (indirizzo **b**, 4 byte): puntatore ai descrittori di trasferimento;
2. **CMD** (indirizzo **b + 4**, 4 byte): registro di comando;
3. **STS** (indirizzo **b + 8**, 4 byte): registro di stato.

La periferica accumula internamente dei byte da una fonte esterna e ogni volta che si scrive il valore 1 nel registro **CMD** cerca di trasferirli tutti in memoria. Non è possibile sapere *a-priori* il numero di byte disponibili all'interno della periferica. I byte verranno trasferiti in una sequenza di zone di memoria descritte da un vettore di *descrittori di trasferimento*, il cui indirizzo è contenuto in **BMPTR**. Ciascun descrittore specifica un indirizzo fisico di partenza e una dimensione. La periferica userà tutte le zone in ordine, fino al trasferimento di tutti i byte disponibili al suo interno. È possibile che le zone non siano sufficienti, nel qual caso i byte in eccesso saranno persi. In ogni caso la periferica invia una richiesta di interruzione al completamento dell'operazione (o perché non ha più byte da trasferire, o perché sono terminate le zone).

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

I descrittori di trasferimento hanno la seguente forma:

```

struct ce_buf_des {
    natl addr;
    natl len;
    natb eod;
    natb eot;
};

```

Prima di avviare una operazione il campo **addr** deve contenere l'indirizzo fisico di una zona di memoria e **len** la sua dimensione in byte; il campo **eod** deve valere 1 se questo è l'ultimo descrittore. Al completamento dell'operazione la periferica scrive in **len** quanti byte della zona ha utilizzato e scrive 1 in **eot** se con questa zona è riuscita a completare il trasferimento di tutti i byte interni.

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 al massimo **quanti** byte dalla periferica numero **id** (tra quelle di questo tipo), copiandoli nel buffer **buf**. La primitiva scrive nel parametro **quanti** il numero di byte effettivamente letti. Inoltre, la primitiva restituisce **true** se il buffer è stato sufficiente a contenere tutti i byte da trasferire, e **false** altrimenti.

È un errore se **buf** non è allineato alla pagina e se **quanti** è zero o è più grande di 10 pagine. In caso di errore la primitiva abortisce il processo chiamante. 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 {
    natw iBMPTR, iCMD, iSTS;
    natl sync;
    natl mutex;
    ce_buf_des buf_des[MAX_CE_BUF_DES];
} __attribute__((aligned(128)));
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 registri BMPTR, STS e RBR, l'indice di un semaforo inizializzato a zero (`sync`), l'indice di un semaforo inizializzato a 1 (`mutex`) e un vettore di descrittori di trasferimento.

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. Durante l'inizializzazione, il registro BMPTR della periferica viene fatto puntare al campo `buf_des` del suo descrittore.

Nota: il modulo sistema mette a disposizione la primitiva

```
addr trasforma(addr ind_virtuale)
```

che restituisce l'indirizzo fisico che corrisponde all'indirizzo virtuale passato come argomento, nello spazio di indirizzamento del processo in esecuzione.