ESERCITAZIONE DMA

Il DMA **da un punto di vista di programmazione del dispositivo è più semplice** di quanto visto per la PIA, USART e PIC, perché dovremo solo inizializzarlo, definendo la modalità di collegamento e poi facendo il trasferimento. Vedremo il trasferimento memoria-memoria e poi memoria-terminale, perché il terminale è un buffer di caratteri, *il DMA potrà in modo agevole prendere tutti i caratteri del buffer e spostarli verso l'area di memoria, nel verso opposto prendere un messaggio in memoria e trasferirlo verso il buffer video del terminale*. Possiamo fare anche esercizi DMA-PIA, la PIA è un dispositivo che fa un trasferimento di un carattere alla volta con handshaking, bisognerebbe mettere in piedi uno scambio di messaggi tra trasmittente e ricevente, un po' contro quello che fa il DMA, che trasferisce blocchi di dati. Per prima cosa, vediamo cosa possiamo programmare del DMA **con Asim, il dispositivo è semplificato perché abbiamo solo 2 canali**, internamente ha 16 registri che sono utilizzati per la configurazione e inizializzazione, in Asim mettiamo il chip DMA in cui inseriamo gli indirizzi di base del dispositivo, finale, il vettore delle interruzioni, la EOP, che va sempre gestita, mentre **in com4 possiamo definire quali sono i dispositivi connessi fisicamente ai due canali**. In nei bit 03 abbiamo l'identificato del dispositivo connesso al primo canale, negli altri due al secondo canale.

Trasferimento memoria-memoria

Abbiamo un'area dati che comincia in 9500 in cui abbiamo inserito il messaggio origine, vogliamo spostarlo nell'area destinazione, che comincia subito dopo. Con le EQU abbiamo definito gli spiazzamenti necessari per accedere ai registri interni del DMA, **ci serve definire l'indirizzo di base del DMA, l'indirizzo del registro che contiene l'indirizzo di partenza** da cui effettuare il trasferimento e **l'indirizzo della destinazione**, verso cui farlo, **il registro contatore dei byte da trasmettere, il registro controllo, di modo e un flag per resettare il DMA**. Poi serve accedere ai flag che qui non useremo ma in generale servono per i trasferimenti con device. Abbiamo definito semplicemente gli spiazzamenti. Nell'area codice facciamo l'inizializzazione definendo la modalità, memoria-memoria e con modalità block. In a0 mettiamo l'indirizzo di base del DMA, poi carichiamo l'indirizzo dell'area origine in caddr0, la destinazione in caddr1, il numero di byte da trasmettere, 34, in ccount.

```
*Area Dati
        ORG         $9500
origine DC.B        '0123456789 messaggio da trasferire'
destinazione    DS.B    34
```

```
dma     EQU     $2010
caddr0      EQU 0
caddr1      EQU 2
ccount1 EQU 3
cntrl       EQU 8
mode        EQU 11
reset       EQU 13
clearmf EQU 14
writeamf    EQU 15
nbyte       EQU 34
* ---------------------------
```

```
        MOVE.W  #dma,A0         ;Carica in A0 l'indirizzo base del DMA

        MOVE.W  #origine,caddr0(A0)     ;Carica l'indirizzo base del blocco sorgente nel
*                                       registro indirizzo del canale 0

        MOVE.W  #destinazione,caddr1(A0)        ;Carica l'indirizzo base del blocco destinazione ne
*                                       registro indirizzo del canale 1

        MOVE.B  #nbyte,ccount1(A0)      ;Carica il numero di byte da spostare nel registro
*                                       conteggio

        MOVE.B  #$90,mode(A0)       ;CANALE 0: setta il trasferimento da memoria ad
*                                       interfaccia, si autoinizializzazione dei registri
*                                       addr e count dopo trasferimento, incremento del
*                                       registro indirizzo,trasferimento in modalità BLOCK

        MOVE.B  #$91,mode(A0)       ;CANALE 1: setta il trasferimento da memoria ad
*                                       interfaccia, si autoinizializzazione dei registri
*                                       addr e count dopo trasferimento, incremento del
*                                       registro indirizzo,trasferimento in modalità BLOCK

        MOVE.B  #$A0,cntrl(A0)          ;CNTRL=1010|0000: abilita trasmissione mem-to-mem e
*                                       abilita il DMA controller (automaticamente viene
*                                       spedita la richiesta di bus al processore)

loop    JMP loop                        ;Loop di attesa del trasferimento, simula un impegno
*                                   fittizio del processore
```

Poi **facciamo il setting tramite il registro di modo, a seconda di un particolare birt che mettiamo a 0 o 1 definiamo come sono configurati i due canali a disposizione**, *con il byte 90 definiamo che sul canale 0 c'è un trasferimento memoria interfaccia, poi a parte andremo a dire che è memoria-memoria,* ***configuriamo separatamente i due canali come memoria-device***. Poi diciamo che *vogliamo l'autoinizializzazione alla fine del trasferimento*, per cui troveremo di nuovo i valori messi in partenza. Vogliamo che *il trasferimento sia a crescere, incrementiamo il registro indirizzo, poi mettiamo il mod block*. La

configurazione sui canali è la stessa e **per discriminare i due canali basta porre un bit a 1**. L'unica cosa che ci fa discriminare il canale è che il primo bit sia posto a 1. Fatto ciò andiamo a inizializzare il registro di controllo, andiamo a scrivere la configurazione in cui abilitiamo la trasmissione memoria-memoria e il DMA controller. **Il byte scritto nel registro di controllo fa partire il trasferimento**, dopo questa istruzione inizia a trasferirsi il messaggio da area origine a destinazione. Per completare la visione del codice, ci resta da vedere **l'interruzione di fine trasferimento, che va sempre gestita**.

```
        ORG $8700

int7        MOVE.L  A0,-(A7)            ;Salva il contesto
            MOVE.W  #dma,A0
            MOVE.B  #0,reset(A0)        ;Resetta il DMA
            MOVE.L  (A7)+,A0          ;Ripristina i registri utilizzati
            RTE
```

In questo caso facciamo il reset del DMA, come operazione, oppure avremmo potuto direttamente mettere l'interruzione no operation e ritorno. Dobbiamo caricare i dispositivi in gioco, tramite la configurazione di Asim, e la memoria nel sistema, perché **andremo opportunamente a settare l'indirizzo della ISR che gestisce questa condizione di EOP**. Caricheremo la memoria con il vettore delle interruzioni specifici, anche qui ***usiamo gli autovettori per cui la EOP sarà collegata materialmente a una delle linee di interruzione del processore.***

Esecuzione in Asim: nel sistema mettiamo solo processore, memoria e DMA, **nelle com mettiamo come gestore delle interruzioni del DMA il processore, su che linea è attaccata EOP e poi servirebbero gli identificativi dei dispositivi attaccati al canale ma ora non ci sono, sono tutti 0**. Per il DMA vediamo tutto il modello di programmazione, abbiamo anche i registri con gli indirizzi e il contatore, che vedremo scorrere di volta in volta. Carichiamo il processore, nella memoria carichiamo la parte della rom che contiene l'indirizzo della ISR legata al segnale EOP.

```
Configuration name: dma.cfg

        CHIP Name: MEMORY
Type: MMU/BUS.   Identif: 01.          BUS: 0000.
Addres 1: 00008000.          Address 2: 00000000.
Com1: 0000.  Com2: 0010.  Com3: 0008.  Com4: 0000.

        CHIP Name: M68000
Type: CPU.       Identif: 02.          BUS: 0001.
Addres 1: 00009000.          Address 2: 00009100.
Com1: 0000.  Com2: 0000.  Com3: 0000.  Com4: 0000.

        CHIP Name: I8237DMA
Type: Device.    Identif: 03.          BUS: 0001.
Addres 1: 00002010.          Address 2: 0000201F.
Com1: 0002.  Com2: 0007.  Com3: 0002.  Com4: 0000.
```

```
dma: I8237DMA 3
                CNTRL:00000000 TEMP:00
MODE0:00000000  RF0:0  MF0:0  MODE1:00000000  RF1:0  MF1:0
   CADDR0:0000    BADDR0:0000  CADDR1:0000    BADDR1:0000
   CCOUNT0:0000 BCOUNT0:0000  CCOUNT1:0000 BCOUNT1:0000
```

Detto questo, per vedere cosa accade ci mettiamo in memoria all'indirizzo in cui partiva l'area di destinazione, all'indirizzo 9522. Eseguiamo il programma, per prima cosa si settano gli indirizzi di partenza negli appositi registri, 9500 come origine, 9522 in destinazione, poi settiamo il modo di entrambi i canali, prima il mod0 e poi mod1, eseguiamo l'istruzione poi che effettivamente abilita il trasferimento, sulla destra (figura in basso, all'indirizzo 9522) vediamo che cominciano ad apparire i caratteri che stiamo trasferendo dall'area di partenza. *Mentre trasferiamo i caratteri il contatore si decrementa, fino a 0 in cui si ha l'interruzione di fine trasferimento*, passiamo a resettare il DMA, si azzerano tutti i registri e si torna al programma principale.

```
dma: I8237DMA 3
                CNTRL:00000000 TEMP:00
MODE0:10010000  RF0:0  MF0:0  MODE1:00000000  RF1:0  MF1:0
   CADDR0:9500    BADDR0:9500  CADDR1:9522    BADDR1:9522
   CCOUNT0:0000 BCOUNT0:0000  CCOUNT1:0022 BCOUNT1:0022

dma: M68000 2
                MOVE.W  #origine,caddr0(A0)          ;Carica l'indiri
*
                MOVE.W  #destinazione,caddr1(A0)             ;Carica
*
                MOVE.B  #nbyte,ccount1(A0)           ;Carica il numer
*
                MOVE.B  #$90,mode(A0)          ;CANALE 0: setta il tras
*
*
*
                MOVE.B  #$91,mode(A0)          ;CANALE 1: setta il tras
*
```

```
dma: I8237DMA 3
                CNTRL:10000000 TEMP:65
MODE0:10010000  RF0:0  MF0:0  MODE1:10010001  RF1:0  MF1:0
   CADDR0:9500    BADDR0:9500  CADDR1:9522    BADDR1:9522
   CCOUNT0:0000 BCOUNT0:0000  CCOUNT1:0022 BCOUNT1:0022

dma: M68000 2
int7            MOVE.L  A0,-(A7)                ;Salva il contes
                MOVE.W  #dma,A0
                MOVE.B  #0,reset(A0)            ;Resetta il DMA
                MOVE.L  (A7)+,A0                ;Ripristina i re
                RTE

                end             start
```

```
dma: MEMORY 1
00009522  30 31 32 33 34 35 36 37 38 39 20 6D 65 73 73 61 67 67 69 6F 20
00009537  64 61 20 74 72 61 66 65 72 69 72 65 00 00 00 00 00 00 00 00 00
0000954C  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00009561  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00009576  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000958B  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000095A0  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000095B5  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000095CA  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000095DF  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000095F4  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00009609  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000961E  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00009633  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00009648  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0000965D  00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

dma.cfg
Configuration name: dma.cfg
```

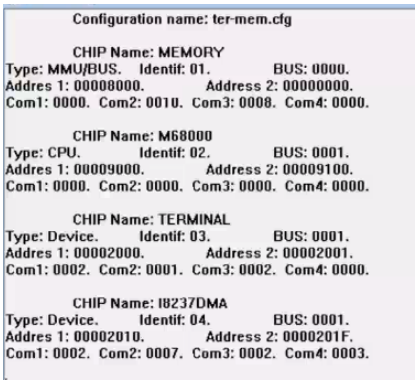
Trasferimento terminale-memoria

Nel trasferiamo dal terminale alla memoria, scriviamo nel terminale e diamo l'avvio del trasferimento, nella realtà il segnale dal device alla memoria chiede di trasferire dei caratteri. Nel nostro caso attiviamo l'interruzione via software, settando opportunamente il request flag: ***usiamo il request flag per far partire la trasmissione***, Mentre nel caso del memoria-memoria bastava settare il registro di controllo e abilitare il trasferimento, qui **dobbiamo realizzare manualmente l'interruzione via software**, per simulare quello che si può fare al limite anche con il segnale reale del dispositivo collegato a un dato canale. Vediamo il codice, inizializziamo il terminale allo stesso modo visto nell'altra esercitazione. In questo caso ***non vediamo le interruzioni gestite con il terminale*** nelle altre esercitazioni, ovvero buffer full e tasto enter, ***perché sono gestite dal processore e non ci interessa***, noi ora vogliamo vedere l'interfacciamento direttamente verso il DMA, senza processore, il processore lo usiamo solo per fare l'inizializzazione di base dei dispositivi. Trasferiamo i dati dal terminale alla memoria, dobbiamo dare l'indirizzo del messaggio, dove vogliamo trasferirlo, in caddr0 e il numero di bye da trasferire nel registro di conteggio, abbiamo messo 5, prende solo 5 dei caratteri che eventualmente scriveremo nel terminale. Ora **settiamo il trasferimento device memoria e lo facciamo accedendo al registro relativo al modo del canale 0 perché colleghiamo il terminale al canale 0**, settiamo il trasferimento device-memoria, poi decidiamo di non autoinizializzare, di fare trasferimento single e incrementare il registro indirizzo, abilitiamo il DMA e poi settiamo a 1 rf0, solo dopo questa istruzione inizia il trasferimento.

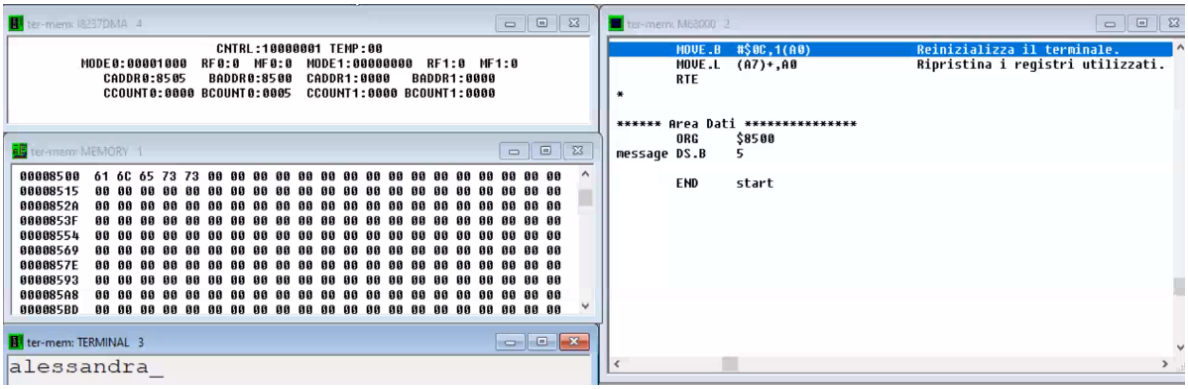
```
* inizializzazione costanti
ter EQU $2000
dma EQU $2010
caddr0  EQU 0
ccount0 EQU 1
cntrl   EQU 8
request EQU 9
mode0   EQU 11
nbyte   EQU 5
*

* programma principale

    ORG $8200

start   MOVE.W  #ter,A0
    MOVE.B  #$3C,1(A0)
*           * programma principale

    ORG $8200

start   MOVE.W  #ter,A0
    MOVE.B  #$3C,1(A0)      Abilita tastiera ,eco e interruzioni; cancella
*                           video e buffer tastiera

    MOVE.W  #dma,A1
    MOVE.B  #nbyte,ccount0(A1)  Carica il numero di byte da spostare nel registro conteggio
    MOVE.W  #message,caddr0(A1)     Carica l'indirizzo di partenza nel registro indirizzo

    MOVE.B  #$08,mode0(A1)      Setta il trasferimento da device a memoria,
*                           no autoinizializzazione dei registri address e count dopo
*                               trasferimento, incremento del registro indirizzo,
*                               trasferimento in modalità SINGLE

    MOVE.B  #$80,cntrl(A1)      Abilita il DMA controller
    MOVE.B  #$08,request(A1)    Accede al flag RF0 e lo setta a 1, per avviare il DMA via SW
*
loop    JMP loop
```

Nel codice c'è anche l'interruzione di fine trasferimento del DMA, che dev'esserci sempre.

Esecuzione in Asim: rispetto a prima **nella com4 mettiamo la 03, identificativo del terminale, colleghiamo il terminale al canale 0**, in questo caso abbiamo anche definito i livelli di interruzione per empty e full ma non ci interessa. Eseguiamo il codice, per prima cosa inizializziamo il terminale, poi facciamo inizializzazioni sul DMA, settiamo il numero di caratteri da trasferire, l'indirizzo del messaggio dove lo posizioniamo 8500, settiamo il modo del canale 0 e abilitiamo il DMA, **prima di poter avviare la richiesta di trasferimento dobbiamo scrivere nel terminale**, scriviamo alessandra, *senza dare enter perché non gestiamo le interruzioni del terminale*, ora attiviamo la richiesta, vediamo A L E S S e poi termina (in esadecimale), finisce e diamo l'interruzione di fine trasferimento, finiamo e ripuliamo il video.




```
******* Interruzione che si attiva alla fine del trasferimento *****
    ORG $8700

int7    MOVE.L  A0,-(A7)    Salva nel SSP i registri utilizzati.
    MOVE.W  #ter,A0
    MOVE.B  #$0C,1(A0)      Reinizializza il terminale.
    MOVE.L  (A7)+,A0        Ripristina i registri utilizzati.
    RTE
*
```

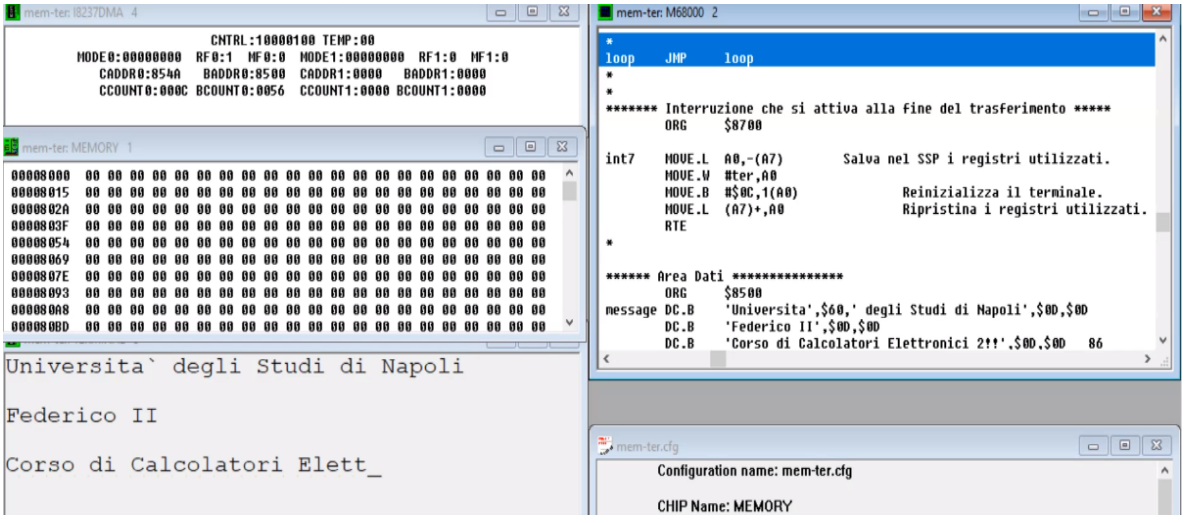


Trasferimento memoria-terminale

Ora mandiamo dalla memoria al terminale, diciamo quanti byte trasferire, dov'è il messaggio e **la modalità di trasferimento sul canale 0, ovvero memoria-device**, è l'unica cosa che cambia, abilitiamo il request flag e visualizziamo il messaggio in area dati, che verrà dato man mano a video. Anche qui abbiamo l'ISR di fine trasferimento.

```
******* Interruzione che si attiva alla fine del trasferimento *****
    ORG $8700

int7    MOVE.L  A0,-(A7)     Salva nel SSP i registri utilizzati.
    MOVE.W  #ter,A0
    MOVE.B  #$0C,1(A0)       Reinizializza il terminale.
    MOVE.L  (A7)+,A0         Ripristina i registri utilizzati.
    RTE
*

****** Area Dati ***************
    ORG $8500
message DC.B    'Universita',$60,' degli Studi di Napoli',$0D,$0D
    DC.B    'Federico II',$0D,$0D
    DC.B    'Corso di Calcolatori Elettronici 2!!',$0D,$0D   86

    END start
```

La configurazione è uguale a quella precedente, il terminale sta sempre sul canale 0, carichiamo la memoria e il processore. Settiamo il numero di byte da trasferire, indirizzo di partenza, abilitiamo il DMA e diamo il request flag: comincia il trasferimento da memoria a terminale, piano piano il contatore scende e visualizziamo i caratteri, alla fine si abilita l'interruzione di fine trasferimento.



Trasferimento PIA-DMA

Con la PIA abbiamo il problema dell'handshaking, **non ci aspettiamo di fare l'handshaking per ogni carattere perché il trasferimento si fa verso il DMA, non c'è qualcuno dall'altro lato che setti segnali ca1 e ca2 che generano gli handshaking**. *È una forzatura collegare DMA e PIA per come sono simulati in Asim*, è solo per fare gli esercizi. Dobbiamo distinguere cosa avviene nella realtà e cosa possiamo simulare. Per far funzionare il tutto **sul sistema che trasmette mettiamo processore memoria e PIA, sul ricevente mettiamo la PIA collegata al DMA, di fatto i dati che arrivano dalla PIA, tramite l'altra, sono spostati in memoria non dal processore** come abbiamo visto, un carattere alla volta tramite l'interruzione: tutti i caratteri che arrivano dalla PIA di s1 a quella di s2 **sono direttamente messi in memoria dal DMA.** Il DMA di s2 trasferisce tutti i caratteri ricevuti dalla PIA direttamente all'area di memoria. Prima la PIA 1 trasferiva, la PIA 2 leggeva e questo dava l'handshaking, il carattere nuovo era trasmesso solo quando crb7 diventava alto, facevamo la lettura fittizia e ricominciavamo. Ora **non ci interessa vedere che la pia2 abbia consumato il dato, non faremo la lettura fittizia, semplicemente in un ciclo scriviamo delle cose nel registro dato della PIA di s1**. Lato s2 tutto quello che viene dalla PIA viene portato dal DMA sulla memoria. Lato trasmittente l'invio è solo un ciclo, una scrittura. Lato ricevente invece andiamo a inizializzare la PIA, in input, usiamo il porto a in ingresso, inizializziamo il DMA, settiamo che in canale 0 deve fare un trasferimento device-memoria e gli attiviamo la richiesta. **Il DMA lavora alla stessa frequenza del processore, trasferisce quello che trova nel registro dato della PIA ad ogni colpo, la PIA è più lenta nel nostro caso ci vogliono 4 colpi per avere un dato nuovo nella PIA**, quindi nella memoria vedremo 4 volte ogni dato.

```
*inizializzazione PIA
    JSR    DVAIN    ;inizializza PI

*inizializzazione DMA

    MOVE.W  SR,D0    ;legge il regis
    ANDI.W  #$D8FF,D0 ;maschera per
    MOVE.W  D0,SR    ;pone liv int a


    MOVE.W  #dma,A1
    MOVE.B  #nbyte,ccount0(A1)  ; C
    MOVE.W  #messaggio,caddr0(A1)
*

    MOVE.B  #$08,mode0(A1)  ; 0000|
*


    MOVE.B  #$80,cntrl(A1)      Abi
    MOVE.B  #$08,request(A1)    ;at
*                       appena si a
*                       periferica
*                       più di una


LOOP    JMP LOOP    ;ciclo caldo
```

```
*************************************
*      PIA1          PIA2
*     _____    _______
*      CA1   |<---------|   CB2
*      CA2   |--------->|   CB1
*     _____|<==== |_______
*      CB1   |<---------|   CA2
*      CB2   |--------->|   CA1
*     _____| ====>|_______
*
*************************************
```

Esecuzione in Asim: vediamo la simulazione Asim, la cosa importante è che **il DMA è come se diventasse gestore delle interruzioni del dispositivo**, è lui a prendere i dati e spostarli in memoria, invece normalmente avremmo dovuto gestire le interruzioni di ricezione del processore e lui avrebbe dovuto spostarli, carattere dopo carattere. Vediamo inizialmente a destra, invio, facciamo le inizializzazioni e poi l'invio, a sinistra diamo un request flag alto, **da quel momento DMA deve prendere quello che trova nel registro dato della PIA e lo mette in memoria, dall'indirizzo 8000**. Appena diamo invio a destra, il DMA sta lavorando e sta trovando dei dati nel registro dato, a destra ci vuole un po' di tempo per scrivere i caratteri mentre a sinistra leggiamo per ogni ciclo di clock, lo stesso numero viene replicato 4 volte perché ci vogliono 4 clock a destra per scrivere nel registro dato. *La replicazione è data alla velocità diversa e alla struttura di istruzioni diverse*. Il sistema reale non funziona così perché la PIA studiata non è fatta per parlare con il DMA, è utile vedere l'esercizio perché si fissa che cambia tutto nella gestione delle interruzioni, del trasferimento che useremmo normalmente. **Quando andiamo da memoria a PIA, la PIA dovrebbe prendere i dati e li dovrebbe dare alla PIA gemella dell'altro sistema ma il problema è che lato trasmittente dovremmo prendere i caratteri dalla memoria, darli alla PIA e da questa darli alla PIA in s2**. **Lato s1 della PIA ci serve sia il porto in input, per prendere dalla memoria, e internamente il dato preso lo dovrebbero dare in uscita a s2, questa cosa non è simulabile in Asim** ma possiamo comunque vedere il codice. Su Asim però vedremo che prende sempre lo stesso dato.

Per ognuna delle 4 periferiche viste dobbiamo conoscere 3 punti di vista, ovvero:

- architettura interna;
- collegamento in termini fisici, con il processore, che di protocollo che si instaura (per inizializzazione o trasferimento a seconda della periferica;
- programmazione del driver.