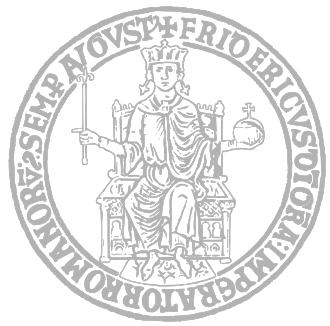


Prova Intercorso 17/05/2022

---

**Computer System Design A.A 2021/2022**  
*Prof. Nicola Mazzocca*



DIETI  
Università Federico II di Napoli

Prova Intercorso - Computer System Design:  
*Simulazione su ASIM*

*Studente:*  
Giuseppe Riccio M63001314

## Indice

<b>1 Traccia d'esame</b>	<b>3</b>
<b>2 Soluzione proposta</b>	<b>4</b>
2.1 Architettura complessiva . . . . .	4
2.2 Protocolli . . . . .	5
2.3 Mappa della memoria . . . . .	5
2.4 Descrizione di alto livello della soluzione tramite pseudocodice . . . . .	6
2.4.1 Schema di funzionamento STANDARD . . . . .	6
2.4.2 Pseudocodice . . . . .	6
2.4.3 Schema di funzionamento FACOLTATIVA . . . . .	7
2.4.4 Passaggio dei parametri sullo stack per il sottoprogramma CHECK . .	8
2.5 Implementazione in Assembly su ASIM . . . . .	9
2.5.1 Configurazione 3nodiPIA.cfg . . . . .	9
2.5.2 Codice Assembly . . . . .	10
2.5.3 Risultati della simulazione . . . . .	19
<b>3 Soluzione modificata con DMA e PIC</b>	<b>21</b>
3.1 Nuova architettura complessiva . . . . .	21
3.2 Protocolli . . . . .	22
3.3 Nuova logica del driver . . . . .	23

## 1 Traccia d'esame

Un sistema è composto da 3 unità, A, B e C, tra loro collegate mediante due periferiche parallele che interconnettono A con B e A con C rispettivamente.

Il sistema opera in due fasi successive come descritto di seguito:

- Fase 1) A riceve un messaggio di N caratteri da B e poi un messaggio di N caratteri da C (**l'ordine è prefissato, non ci sono sovrapposizioni fra B e C nella prima fase**); dopo averli ricevuti, A verifica se i messaggi ricevuti da B e C sono uguali invocando una subroutine CHECK (*LA SUBROUTINE NON DEVE ESSERE IMPLEMENTATA MA SOLO INVOCATA PASSANDO OPPORTUNAMENTE I PARAMETRI SULLO STACK*);
- Fase 2) Al termine della fase 1, se i messaggi sono uguali, A riceverà T ulteriori messaggi (T fissato a scelta dello studente), ricevendo prima un intero messaggio da B e poi un intero messaggio da C in maniera alternata. Se invece i messaggi ricevuti nella fase 1 sono diversi, A riceverà T ulteriori messaggi, ricevendo prima un messaggio da C e poi da B, in maniera alternata.

**IPOTESI FACOLTATIVA:** nella fase 1 è possibile rimuovere l'ipotesi che l'ordine di arrivo dei messaggi sia prefissato, e gestire l'arrivo di caratteri anche intervallati da B e da C. Si discuta **almeno qualitativamente** questa ipotesi (come cambierebbe il flusso di controllo). **Lo studente può anche scegliere di implementare il sistema A rispetto alla sola ipotesi facoltativa.**

Dopo aver sviluppato l'intero progetto, si illustri come cambierebbero l'architettura complessiva e la logica del driver se venisse inserito un componente DMA per la comunicazione fra A e i nodi B e C.

**Nota:** non è richiesta l'implementazione completa di un nuovo programma, ma lo studente dovrà indicare schematicamente le principali modifiche necessarie al codice assembly già prodotto (è preferibile a tale scopo indicare a parte gli stralci di codice da inserire ove necessario).

## 2 Soluzione proposta

### 2.1 Architettura complessiva

L'architettura complessiva del sistema 1 presenta 3 nodi, il nodo A ha una CPU, una memoria e due periferiche parallele PIA connesse rispettivamente una al nodo B ed un'altra al nodo C. I nodi B e C sono identici in termini di componenti presenti nell'architettura, infatti, entrambi presentano una CPU, una memoria ed una periferica parallela PIA.

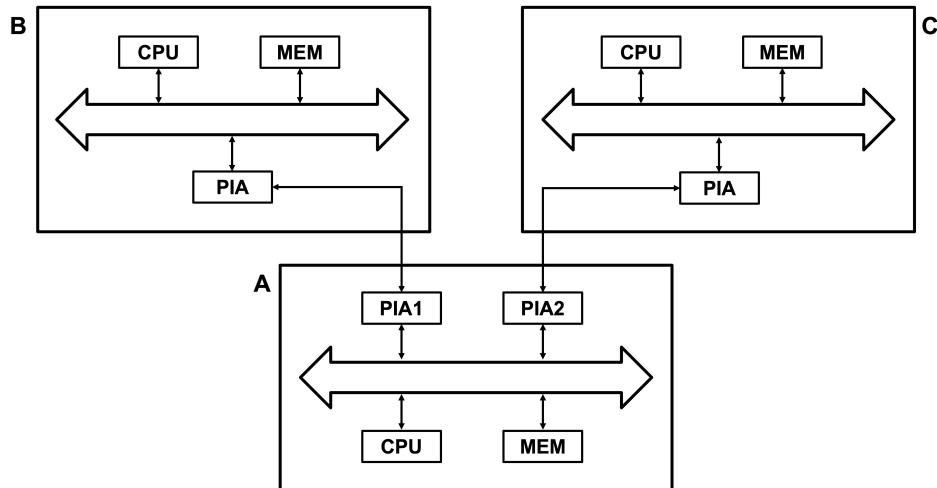


Figura 1: Architettura complessiva del sistema

Per quanto riguarda le periferiche parallele del nodo A, esse sono collegate alle linee di interruzione del processore rispettivamente sulla 3 e sulla 4 linea, quindi, all'arrivo di un'interruzione il processore andrà ad individuare nel vettore delle interruzioni la relativa ISR da invocare per la gestione di tale periferica. Ad esempio per la PIA1, il livello di interruzione è il 3, dunque, il processore andrà nella tabella e sommerà alla base \$64 l'offset 3 il cui risultato è \$6C dove si trova esattamente la ISRB, come si nota dalla Figura 2.

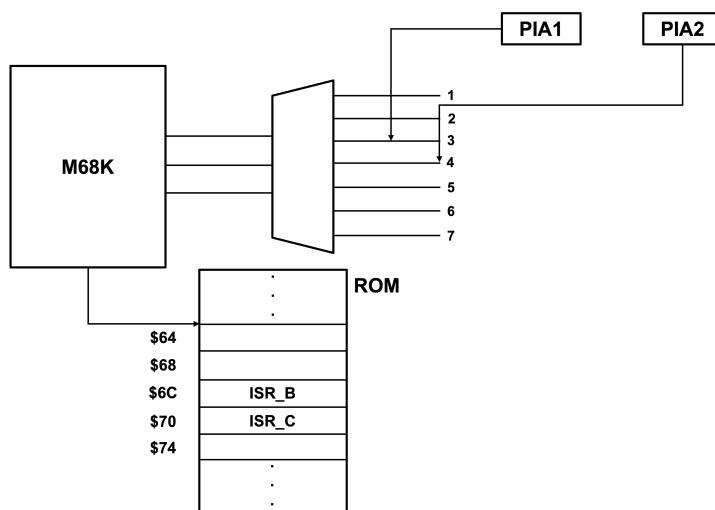


Figura 2: Livello delle interruzioni del nodo A

## 2.2 Protocolli

Per quanto riguarda i protocolli usati nel sistema ci riferiamo, a quello della PIA, in particolare nella sua configurazione handshaking '100' 3, la periferica parallela si comporta nel modo seguente:

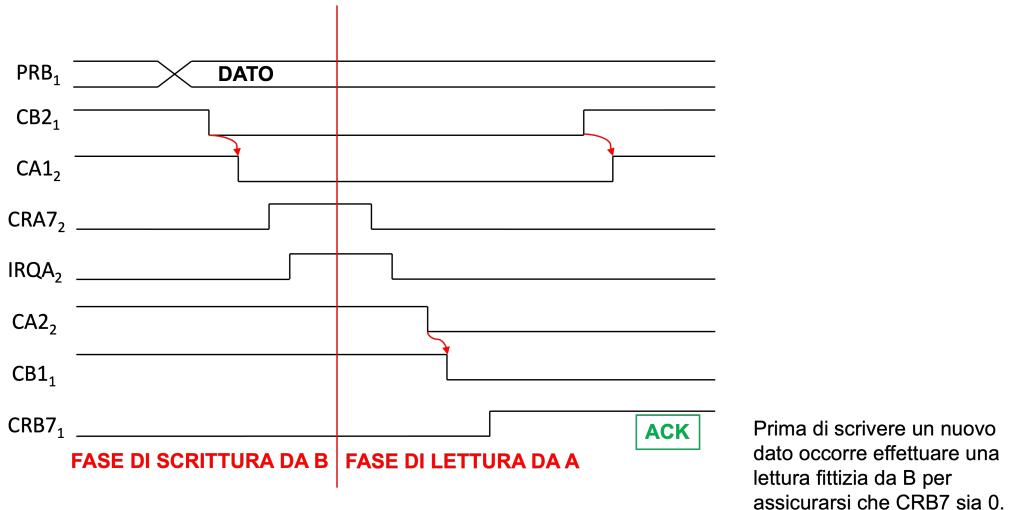


Figura 3: Protocollo 100 handshaking della PIA

## 2.3 Mappa della memoria

La memoria del sistema in questione, dovrà contenere oltre ai dati per la gestione dei messaggi provenienti dai 2 nodi, anche le informazioni relative al programma MAIN ed alle ISR per la gestione delle interruzioni sulle periferiche PIA 4.

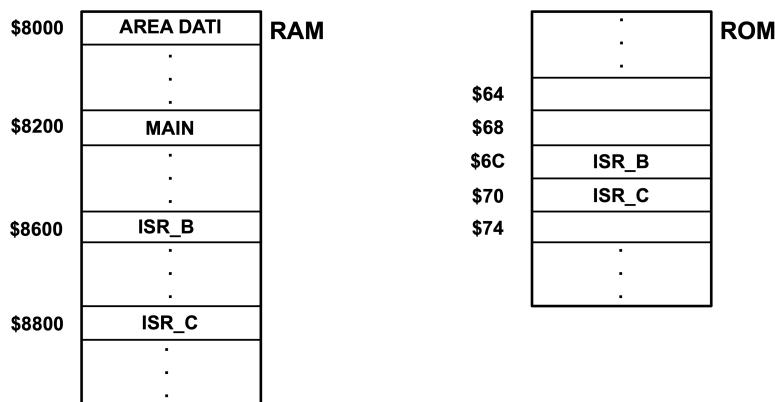


Figura 4: Mappa della memoria

## 2.4 Descrizione di alto livello della soluzione tramite pseudocodice

#### 2.4.1 Schema di funzionamento STANDARD

Nell'ipotesi STANDARD, il sistema dovrebbe funzionare come descritto in Figura 5, in cui nella FASE 1, l'ordine di arrivo dei messaggi è fissato a priori e dunque non possono esserci sovrapposizioni, mentre la FASE 2 può seguire due schemi in base al valore dei 2 messaggi ricevuti durante la FASE 1.



Figura 5: Ipotesi di funzionamento STANDARD

### 2.4.2 Pseudocodice

Lo pseudocodice per implementare il comportamento appena illustrato è il seguente:

**MAIN:** Inizializzo le 2 PIA in ricezione con interrupt.  
Passo allo stato USER e abilito le interruzioni.  
Ciclo caldo.

```

ISR_B: if (FASE1 == 1){
    leggo carattere da B
    cont_B++
    if (cont_B == N){
        cont_B = 0
    }
} else {
    if (UGUALI == 1){
        if (TURNO_B == 1){
            leggo carattere da B
            cont_B++
            if (cont_B == N){
                cont_B = 0
                TURNO_B = 0
                if (sospC == 1){
                    risveglia_C
                    NOTA: Nella funzione risveglia_C ,
                           oltre alla lettura del carattere
                           da C, viene anche abbassato il
                           flag di sospC.
                }
            }
        } else {sospB = 1}
    }
}

```

```

    } else {
        NOTA: In questo ramo else , si entra se i due
        messaggi ricevuti nella FASE1 sono diversi ,
        quindi , si effettuano le stesse operazioni
        viste nel ramo if (UGUALI == 1) con l'unica
        modifica che al posto della verifica (TURNO_B
        == 1) occorre verificare (TURNO_C == 0) ed
        alla fine mettere TURNO_C = 1 dopo la
        ricezione del messaggio .
    }
}

ISR_C: if (FASE1 == 1){
    leggo carattere da C
    cont_C++
    if (cont_C == N){
        cont_C = 0
        passo parametri sullo STACK
        salto al sottoprogramma CHECK
        FASE1 = 0
    }
} else {
    if (UGUALI == 1){
        if (TURNO_B == 0){
            leggo carattere da C
            cont_C++
            if (cont_C == N){
                cont_C = 0
                TURNO_B = 1
                if (sospB == 1){
                    risveglia_B
                    NOTA: Nella funzione risveglia_B ,
                    oltre alla lettura del carattere
                    da B, viene anche abbassato il
                    flag di sospB .
                }
            }
        } else {sospC = 1}
    } else {
        NOTA: In questo ramo else eseguo le stesse
        operazioni viste nel ramo if , sostituendo
        if (TURNO_B == 0) con if (TURNO_C == 1) e
        settando al termine della ricezione del
        messaggio TURNO_C = 0.
    }
}

```

#### 2.4.3 Schema di funzionamento FACOLTATIVA

Nell'ipotesi FACOLTATIVA, il sistema dovrebbe funzionare come descritto in Figura 6, in cui nella FASE 1, l'ordine di arrivo dei messaggi non è prefissato e dunque possono esserci

sovraposizioni. Occorre, quindi, gestire l'alternanza nell'arrivo dei caratteri dai nodi B e C, mentre la FASE 2 non cambia rispetto all'ipotesi STANDARD.

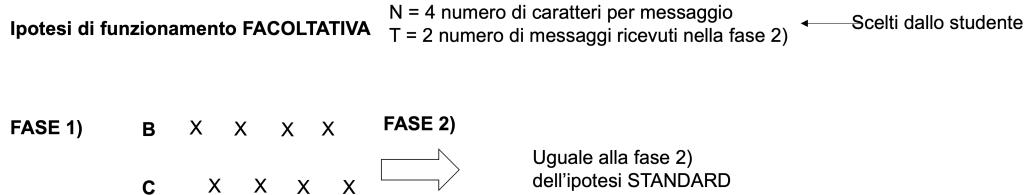


Figura 6: Ipotesi di funzionamento FACOLTATIVA

Per gestire l'alternanza dei caratteri nella FASE 1

#### 2.4.4 Passaggio dei parametri sullo stack per il sottoprogramma CHECK

Per invocare il sottoprogramma che confronta se i 2 messaggi ottenuti nella FASE 1 dal nodo A sono uguali, occorre passare ad esso i parametri necessari a tale scopo. In particolare, si è scelto di usare le variabili mostrate in Figura 7.

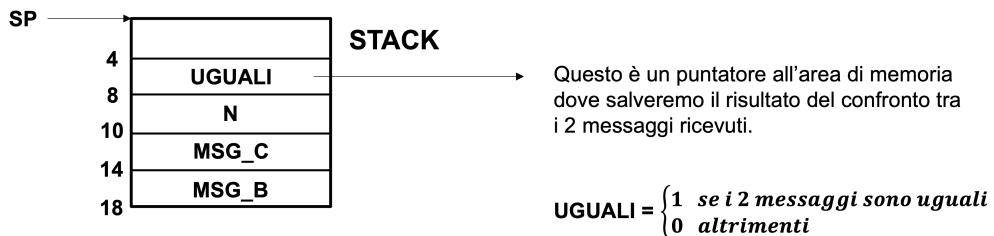


Figura 7: Stack per il passaggio dei parametri alla subroutine CHECK

## 2.5 Implementazione in Assembly su ASIM

### 2.5.1 Configurazione 3nodiPIA.cfg

```
Configuration name: 3nodi_PIA.cfg

CHIP Name: M68000
Type: CPU. Identif: 01. BUS: 0002.
Address 1: 00009000. Address 2: 00009200.
Com1: 0000. Com2: 0000. Com3: 0000. Com4: 0000.

CHIP Name: Memory
Type: MMU/BUS. Identif: 02. BUS: 0000.
Address 1: 00008000. Address 2: 00000000.
Com1: 0000. Com2: 0010. Com3: 0008. Com4: 0000.

CHIP Name: M6821PIA
Type: Device. Identif: 03. BUS: 0002.
Address 1: 00002004. Address 2: 00002007.
Com1: 0001. Com2: 0003. Com3: 0004. Com4: 0209.

CHIP Name: M68000
Type: CPU. Identif: 04. BUS: 0005.
Address 1: 00009000. Address 2: 00009200.
Com1: 0000. Com2: 0000. Com3: 0000. Com4: 0000.

CHIP Name: Memory
Type: MMU/BUS. Identif: 05. BUS: 0000.
Address 1: 00008000. Address 2: 00000000.
Com1: 0000. Com2: 0010. Com3: 0008. Com4: 0000.

CHIP Name: M6821PIA
Type: Device. Identif: 06. BUS: 0005.
Address 1: 00002004. Address 2: 00002007.
Com1: 0004. Com2: 0003. Com3: 0004. Com4: 020A.

CHIP Name: M68000
Type: CPU. Identif: 07. BUS: 0008.
Address 1: 00009000. Address 2: 00009200.
Com1: 0000. Com2: 0000. Com3: 0000. Com4: 0000.

CHIP Name: Memory
Type: MMU/BUS. Identif: 08. BUS: 0000.
Address 1: 00008000. Address 2: 00000000.
Com1: 0000. Com2: 0010. Com3: 0008. Com4: 0000.

CHIP Name: M6821PIA
Type: Device. Identif: 09. BUS: 0008.
Address 1: 00002004. Address 2: 00002007.
Com1: 0007. Com2: 0003. Com3: 0004. Com4: 0203.

CHIP Name: M6821PIA
Type: Device. Identif: 0A. BUS: 0008.
Address 1: 00002008. Address 2: 0000200B.
Com1: 0007. Com2: 0004. Com3: 0005. Com4: 0206.
```

Figura 8: Configurazione 3 nodi con PIA su ASIM

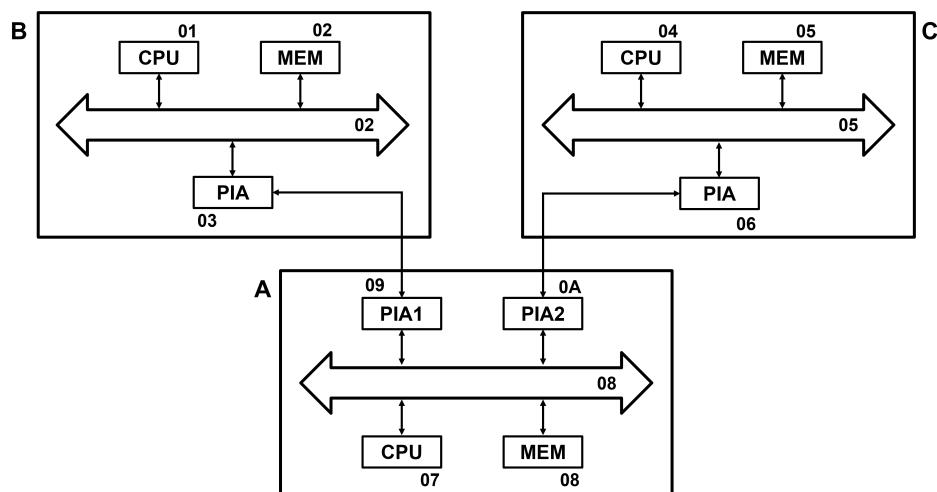


Figura 9: Architettura complessiva con identificativi ASIM

## 2.5.2 Codice Assembly

Iniziamo con il presentare il codice Assembly dei 2 nodi (B e C) che effettuano la trasmissione dei messaggi al nodo A. Di seguito è illustrato il codice del nodo B:

```
*****NODO B - TX*****
*AREA DATI
    ORG      $8000
MES      DC.B     1,2,3,4,5,6,7,8,9,10,11,12
DIM      DC.B     12
CNTB    DS.B     1

PIADB   EQU      $2006
PIACB   EQU      $2007

*AREA MAIN
    ORG      $8200
MAIN    JSR      INIZIALIZZA

MOVEA.L #PIACB,A1          ; indirizzo registro di controllo CRB
MOVEA.L #PIADB,A2          ; indirizzo registro PRB
MOVEA.L #MES,A0 ; indirizzo area messaggio

MOVE.W SR,D1
ANDI.W #$D8FF,D1          *maschera per le interruzioni.
MOVE.W D1,SR

* invio primo carattere:
INVIO1 MOVE.B (A2),D1      *lettura fittizia da PRB
MOVE.B (A0),(A2)

MOVE.B #1,CNTB

LOOP    JMP      LOOP

INIZIALIZZA      MOVE.B #0,PIACB
                  MOVE.B #$FF,PIADB
                  MOVE.B #%00100101,PIACB
RTS

*interruzione livello 4 mappato a $70
    ORG      $8800
INT4    MOVE.L  A0,-(A7)
                  MOVE.L  A1,-(A7)
                  MOVE.L  D0,-(A7)
                  MOVE.L  D1,-(A7)

CLR      D0
CLR      D1
```

```

MOVEA.L #PIADB,A0
MOVEA.L #MES,A1
MOVE.B CNTB,D0

*Lettura fittizia
MOVE.B (A0),D1           *Così posso riusare D1

MOVE.B DIM,D1   *Dimensione totale dei byte da inviare

CMP.B      D0,D1           *un controllo su D1
BEQ        RIPRISTINO

ADDA       D0,A1
MOVE.B    (A1),(A0)
ADD.B      #1,D0
CMP.B      D0,D1
BNE        FUORI

FUORI     MOVE.B D0,CNTB

RIPRISTINO MOVE.L (A7)+,D1
MOVE.L (A7)+,D0
MOVE.L (A7)+,A1
MOVE.L (A7)+,A0

RTE
END      MAIN

```

Il codice del nodo C è del tutto simile, come si può notare nel seguente frammento di codice:

\*\*\*\*\*NODO C – TX\*\*\*\*\*

```

*AREA DATI
ORG      $8000
MES      DC.B   1,2,3,4,5,6,7,8,9,10,11,12
DIM      DC.B   12
CNTC     DS.B   1

PIADC    EQU    $2006
PIACC    EQU    $2007

*AREA MAIN
ORG      $8200
MAIN    JSR    INIZIALIZZA

MOVEA.L #PIACC,A1      ;indirizzo registro di controllo CRB
MOVEA.L #PIADC,A2      ;indirizzo registro PRB
MOVEA.L #MES,A0 ;indirizzo area messaggio

MOVE.W SR,D1

```

```

ANDI.W #$D8FF,D1      *maschera per le interruzioni .
MOVE.W D1,SR

* invio primo carattere :
INVIO1 MOVE.B (A2),D1      *lettura fittizia da PRB
MOVE.B (A0),(A2)

MOVE.B #1,CNTC

LOOP    JMP    LOOP

INIZIALIZZA   MOVE.B #0,PIACC
               MOVE.B #$FF,PIADC
               MOVE.B #%%00100101,PIACC
RTS

*interruzione livello 4 mappato a $70
ORG      $8800
INT4    MOVE.L A0,-(A7)
        MOVE.L A1,-(A7)
        MOVE.L D0,-(A7)
        MOVE.L D1,-(A7)

CLR      D0
CLR      D1
MOVEA.L #PIADC,A0
MOVEA.L #MES,A1
MOVE.B CNTC,D0

*Lettura fittizia
MOVE.B (A0),D1      *Così posso riusare D1

MOVE.B DIM,D1      *Dimensione totale dei byte da inviare

CMP.B      D0,D1      *un controllo su D1
BEQ       RIPRISTINO

ADDA      D0,A1
MOVE.B (A1),(A0)
ADD.B      #1,D0
CMP.B      D0,D1
BNE       FUORI

FUORI    MOVE.B D0,CNTC

RIPRISTINO MOVE.L (A7)+,D1
MOVE.L (A7)+,D0
MOVE.L (A7)+,A1
MOVE.L (A7)+,A0

```

RTE  
END      MAIN

Il nodo A nella prima fase riceve un messaggio di N caratteri da B e poi un messaggio di N caratteri da C, dopo averli ricevuti, A verifica se i messaggi ricevuti da B e C sono uguali invocando una subroutine CHECK. Al termine della FASE 1, se i messaggi sono uguali, A riceverà T ulteriori messaggi (T fissato a scelta dello studente), ricevendo prima un intero messaggio da B e poi un intero messaggio da C in maniera alternata. Se invece i messaggi ricevuti nella FASE 1 sono diversi, A riceverà T ulteriori messaggi, ricevendo prima un messaggio da C e poi da B, in maniera alternata. Tale nodo A, dunque presenta due dispositivi PIA, che si comportano da ricevitori. Una PIA genera una interruzione di livello 3, mentre l'altra di livello 4. Dunque, seguendo lo pseudocodice 2.4.2 visto in precedenza si è passati all'implementazione delle due ISR di gestione delle ricezioni dei messaggi da B e C, come di seguito:

\*\*\*\*\*NODO A – RX\*\*\*\*\*

\*AREA DATI

	ORG	\$8000	
MSGB	DS.B	12	*Appoggio per il messaggio ricevuto dal primo tx
MSGC	DS.B	12	*appoggio per il messaggio ricevuto dal secondo tx
CNTB	DC.B	0	*numero di byte ricevuti da B
CNTC	DC.B	0	*numero di byte ricevuti da C
FASE1	DC.B	1	*flag che segnala se ci troviamo nella fase 1 (=1) *oppure nella fase 2
N	EQU	4	*numero di caratteri che formano un messaggio
T	EQU	2	*numero di messaggi ricevuti nella fase 2
UGUALI	DS.B	1	*Area di memoria per il salvataggio dell'operazione *di confronto dei 2 messaggi ricevuti nella fase 1
TURNOB	DC.B	1	*flag per segnalare che e' il turno di B
TURNOC	DC.B	1	*flag per segnalare che e' il turno di C
SOSPB	DC.B	0	*flag per segnalare che B e' sospeso
SOSPC	DC.B	0	*flag per segnalare che C e' sospeso
PIADA	EQU	\$2004	
PIACA	EQU	\$2005	
PIADA2	EQU	\$2008	
PIACA2	EQU	\$2009	

\*AREA MAIN

ORG      \$8200  
MAIN     JSR    INIZIALIZZA

MOVE.W SR,D1  
ANDI.W #\$D8FF,D1  
MOVE.W D1,SR

LOOP    JMP    LOOP

```

INIZIALIZZA    MOVE.B #0,PIACA
                MOVE.B #0,PIADA
                MOVE.B #%00100101,PIACA

                MOVE.B #0,PIACA2
                MOVE.B #0,PIADA2
                MOVE.B #%00100101,PIACA2
                RTS

*****SOTOPROGRAMMA CHECK*****
*           STACK
*           A1          (4 BYTE)
*           A0          (4 BYTE)
*           D1          (4 BYTE)
*           D0          (4 BYTE)
*           A6          (4 BYTE) <---A6
*           IND. RITORNO (4 BYTE)
*           UGUALI      (4 BYTE)
*           N           (2 BYTE)
*           MSGC        (4 BYTE)
*           MSGB        (4 BYTE)

CHECK     LINK   A6,#-16
          MOVE.L D0,-4(A6)
          MOVE.L D1,-8(A6)
          MOVE.L A0,-12(A6)
          MOVE.L A1,-16(A6)

          MOVE.L 8(A6),A0      *UGUALI
          MOVE.W 12(A6),D0      *N
          MOVE.L 14(A6),A2      *MSGC
          MOVE.L 18(A6),A1      *MSGB

          MOVE.B (A0),D1

CONF      MOVE.B (A1),D2
          CMP.B  (A2),D2
          BNE    EXIT           *se i caratteri sono diversi
                           *esco dalla funzione
          ADDA   #1,A1
          ADDA   #1,A2
          ADD.B  #1,D3
          CMP.B  D3,D0
          BNE    CONF           *ci sono ancora caratteri da
                           *confrontare? se si continuo
                           *a confrontare

```

```

MOVE.B #1,D1           *sono arrivato a confrontare
          *tutti i caratteri,
          *quindi i messaggi sono
          *uguali (=1)
MOVE.B D1,( A0)

EXIT    MOVE.L -4(A6),D0
        MOVE.L -8(A6),D1
        MOVE.L -12(A6),A0
        MOVE.L -16(A6),A1
        UNLK      A6
        RTS

*****Interrupt livello 3 --> PIA1 da B
ORG     $8600

ISRB    MOVEM.L D0-D7/A0-A6,-(A7)
        MOVEA.L #PIADA,A0      *sposto in A0 indirizzo PIADA
        MOVEA.L #MSGB,A1      *sposto in A1 indirizzo MSGB
        MOVE.B #N,D0          *sposto in D0 il numero di caratteri N
        MOVE.B CNTB,D1 *sposto in D1 contatore di caratteri da B
        CMP.B #1,FASE1        *sono nella fase 1? se si , prosegue
        BNE      FASE2B

        CMP.B #1,TURNOB       *nella fase 1 secondo l'ipotesi
                               *STANDARD l'ordine di arrivo
                               *dei messaggi e' prefissato
                               *quindi ricevo prima da B
        BNE      F1B

        MOVE.B (A0),(A1,D1)   *leggo carattere da B
        ADDI.B #1,D1          *incremento il contatore di B
        MOVE.B D1,CNTB
        CMP.B D0,D1          *verifico se ho ricevuto N caratteri
        BLT      ESCB          *se no , esco

        MOVE.B #0,CNTB
        MOVE.B #0,TURNOB

        JMP      ESCB

F1B     MOVE.B #1,SOSPB
        CMP.B #1,SOSPC
        BNE      ESCB
        MOVEM.L D0-D7/A0-A6,-(A7)
        JSR      RISVEGLIAC
        MOVEM.L (A7)+,D0-D7/A0-A6
        JMP      ESCB

FASE2B CMP.B #1,UGUALI      * i messaggi ricevuti nella

```

```

*prima fase sono uguali?
*se si , proseguo
BNE      FUNB

CMP.B    #1,TURNOB      *verifico se e' il turno di B
BNE      SOSPENDIB

MOVE.B   (A0),(A1,D1)   *leggo carattere da B
ADDI.B   #1,D1          *incremento il contatore di B
MOVE.B   D1,CNTB
CMP.B    D0,D1          *verifico se ho ricevuto N caratteri
BLT     ESCB            *se no , esco

MOVE.B   #0,CNTB
MOVE.B   #0,TURNOB
CMP.B    #1,SOSPC       *per caso nel frattempo C ha
                        *interrotto e si e' sospeso?
                        *se si , servo la sua richiesta
BNE      ESCB

MOVEM.L  D0-D7/A0-A6,-(A7)
JSR      RISVEGLIAC
MOVEM.L  (A7)+,D0-D7/A0-A6
JMP      ESCB

FUNB     CMP.B    #0,TURNOC      *verifico che non sia il turno di C
BNE      SOSPENDIB

MOVE.B   (A0),(A1,D1)   *leggo carattere da B
ADDI.B   #1,D1          *incremento il contatore di B
MOVE.B   D1,CNTB
CMP.B    D0,D1          *verifico se ho ricevuto N caratteri
BLT     ESCB            *se no , esco

MOVE.B   #0,CNTB
MOVE.B   #1,TURNOC
CMP.B    #1,SOSPC       *per caso nel frattempo C ha
                        *interrotto e si e' sospeso?
                        *se si , servo la sua richiesta
BNE      ESCB

MOVEM.L  D0-D7/A0-A6,-(A7)
JSR      RISVEGLIAC
MOVEM.L  (A7)+,D0-D7/A0-A6
JMP      ESCB

SOSPENDIB      MOVE.B  #1,SOSPB

ESCB      MOVEM.L (A7)+,D0-D7/A0-A6
RTE

```

```
***** Interrupt livello 4 —> PIA2 da C
ORG      $8800

ISRC    MOVE.M.L D0-D7/A0-A6,-(A7)
        MOVEA.L #PIADA2,A0          *sposto in A0 indirizzo PIADA2
        MOVEA.L #MSGC,A1          *sposto in A1 indirizzo MSGC
        MOVE.B #N,D0              *sposto in D0 il numero di caratteri N
        MOVE.B CNTC,D1 *sposto in D1 contatore di caratteri da C
        CMP.B #1,FASE1           *sono nella fase 1? se si , prosegua
        BNE      F2C

        CMP.B #0,TURNOB          *nella fase 1 secondo l'ipotesi
                                *STANDARD l'ordine di arrivo dei
                                *messaggi e' prefissato quindi
                                *ricevo da C solo dopo che ho
                                *ricevuto da B
        BNE      SOSC

        MOVE.B (A0),(A1,D1)       *leggo carattere da C
        ADDI.B #1,D1             *incremento il contatore di C
        MOVE.B D1,CNTC
        CMP.B D0,D1              *verifico se ho ricevuto N caratteri
        BLT      ESCC             *se no , esco

        MOVE.B #0,CNTC
        MOVE.B #1,TURNOB

        MOVE.L #MSGB,-(A7)
        MOVE.L #MSGC,-(A7)
        MOVE.W #N,-(A7)
        MOVE.L #UGUALI,-(A7)

        JSR      CHECK            *chiamo la subroutine per verificare
                                *se i 2 messaggi ricevuti sono uguali

        ADDA   #14,A7
        MOVE.B #0,FASE1

        CMP.B #1,UGUALI
        BNE      ESCC
        MOVE.M.L D0-D7/A0-A6,-(A7)
        JSR      RISB
        MOVE.M.L (A7)+,D0-D7/A0-A6

        JMP      ESCC

F2C    CMP.B #1,UGUALI         * i messaggi ricevuti nella prima
                                *fase sono uguali? se si , prosegua
        BNE      FC
```

```

CMP.B #0,TURNOB      *verifico che non sia il turno di B
BNE SOSC

MOVE.B (A0),(A1,D1)   *leggo carattere da C
ADDI.B #1,D1          *incremento il contatore di C
MOVE.B D1,CNTC
CMP.B D0,D1          *verifico se ho ricevuto N caratteri
BLT ESCC              *se no, esco

MOVE.B #0,CNTC
MOVE.B #1,TURNOB
CMP.B #1,SOSPB        *per caso nel frattempo B ha
                      *interrotto e si e' sospeso?
                      *se si, servo la sua richiesta
BNE ESCC

MOVEM.L D0-D7/A0-A6,-(A7)
JSR RISB
MOVEM.L (A7)+,D0-D7/A0-A6
JMP ESCC

FC    CMP.B #1,TURNOC      *verifico che sia il turno di C
      BNE SOSC

MOVE.B (A0),(A1,D1)   *leggo carattere da C
ADDI.B #1,D1          *incremento il contatore di C
MOVE.B D1,CNTC
CMP.B D0,D1          *verifico se ho ricevuto N caratteri
BLT ESCC              *se no, esco

MOVE.B #0,CNTC
MOVE.B #0,TURNOC
CMP.B #1,SOSPB        *per caso nel frattempo B ha
                      *interrotto e si e' sospeso?
                      *se si, servo la sua richiesta
BNE ESCC

MOVEM.L D0-D7/A0-A6,-(A7)
JSR RISB
MOVEM.L (A7)+,D0-D7/A0-A6
JMP ESCC

SOSC MOVE.B #1,SOSPC

ESCC MOVEM.L (A7)+,D0-D7/A0-A6
      RTE

RISB MOVE.B #0,SOSPB      *abbasso il flag di sospeso di B
      MOVEA.L #PIADA,A0
      MOVEA.L #MSGB,A1
      MOVE.B #N,D0

```

```

MOVE.B CNTB,D1

MOVE.B (A0),(A1,D1) *leggo carattere da B
ADDI.B #1,D1 *incremento il contatore di B
MOVE.B D1,CNTB
CMP.B D0,D1 *verifico se ho ricevuto N caratteri
BLT USC *se no, esco

MOVE.B #0,CNTB
USC RTS

RISVEGLIAC MOVE.B #0,SOSPC *abbasso il flag di sospeso di C
MOVEA.L #PIADA2,A0
MOVEA.L #MSGC,A1
MOVE.B #N,D0
MOVE.B CNTC,D1

MOVE.B (A0),(A1,D1) *leggo carattere da C
ADDI.B #1,D1 *incremento il contatore di C
MOVE.B D1,CNTC
CMP.B D0,D1 *verifico se ho ricevuto N caratteri
BLT ESCI *se no, esco

MOVE.B #0,CNTC
ESCI RTS

END MAIN

```

### 2.5.3 Risultati della simulazione

Una volta caricato il sistema con la configurazione, l’assembler e le ISR, la memoria del nodo A si presenta come in Figura 10, in particolare, abbiamo che il flag della FASE 1 è pari ad 1, così come i flag che indicano il turno dei nodi B e C.

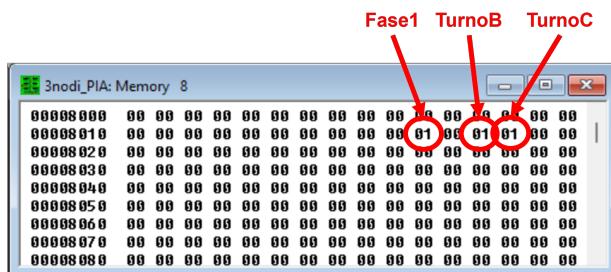


Figura 10: Memoria del sistema all’inizio

Simuliamo il primo caso, ovvero, quello in cui nella FASE 1 abbiamo ricevuto due messaggi uguali infatti nella Figura 11 possiamo notare come al termine della FASE 1 abbiamo in memoria i due messaggi. Essendo che i due messaggi sono uguali, abbiamo che la corrispondente variabile è settata ad 1 dalla funzione CHECK, la FASE 1 è dunque terminata (si noti valore pari a 0).

00008000	01	02	03	04	00	00	00	00	00	00	01	02	03	04
00008010	00	00	00	00	00	00	00	00	00	00	01	01	01	01
00008020	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008030	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008040	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008050	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008060	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008070	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008080	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Figura 11: Fase 1 - Caso di messaggi uguali

Dato che i messaggi sono uguali, quindi, nella FASE 2 ricevo prima un messaggio da B e poi da C ciò è possibile sospendendo C quando ricevo da B e viceversa, Figura 12.

00008000	05	06	07	08	00	00	00	00	00	00	01	02	03	04
00008010	00	00	00	00	00	00	00	00	00	00	01	01	01	01
00008020	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008030	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008040	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008050	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008060	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008070	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008080	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Figura 12: Fase 2 - Caso di messaggi uguali

L'altro caso è quello in cui nella FASE 1 ricevo 2 messaggi diversi, in questo caso uguali è pari a 0, Figura 13.

00008000	04	03	02	01	00	00	00	00	00	00	01	02	03	04
00008010	00	00	00	00	00	00	00	00	00	00	01	01	01	01
00008020	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008030	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008040	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008050	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008060	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008070	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00008080	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Figura 13: Fase 1 - Caso di messaggi diversi

Quindi, quando i 2 messaggi sono diversi come da traccia, ricevo prima un messaggio da C e poi uno da B, sfruttando come nel caso precedente i flag di sospeso, Figura 14.

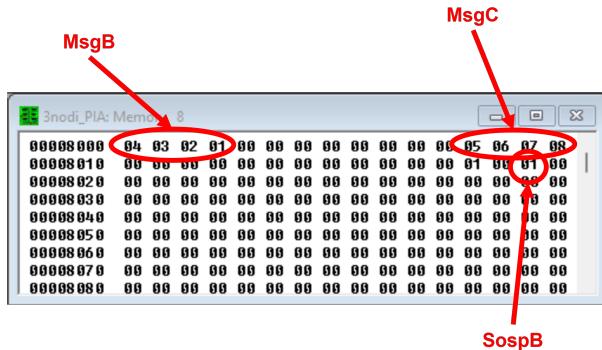


Figura 14: Fase 2 - Caso di messaggi diversi

### 3 Soluzione modificata con DMA e PIC

#### 3.1 Nuova architettura complessiva

La nuova architettura complessiva del sistema con DMA e PIC diventa:

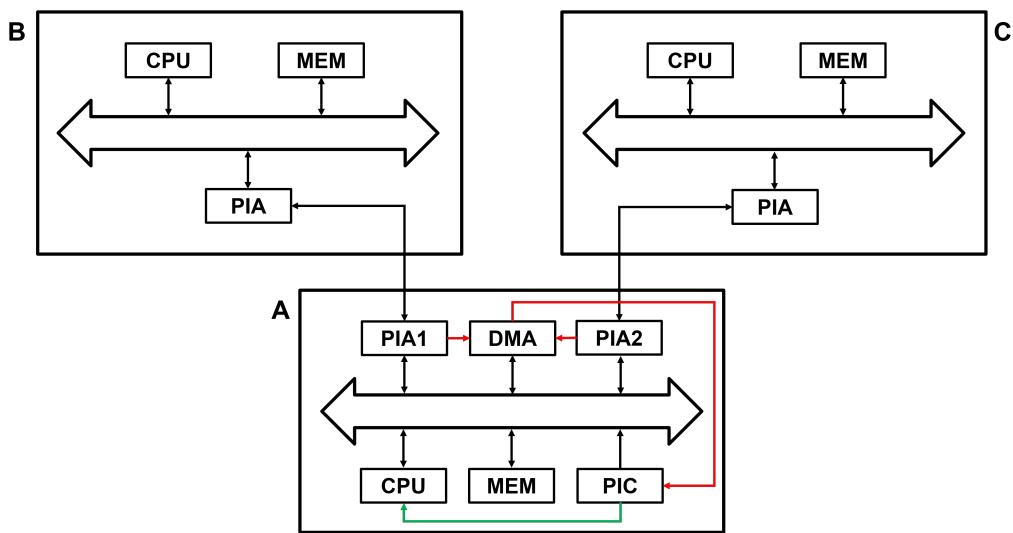


Figura 15: Architettura complessiva del sistema con DMA e PIC

In particolare, sul canale 0 si collega la PIA del nodo B e sul canale 1 si collega la PIA del nodo C. Tuttavia, occorre notare che essendo i 2 canali del DMA entrambi usati potrebbero nascere dei conflitti, che occorre risolvere adottando le politiche di priorità del DMA stesso, che può assegnare o una priorità fissa (ad esempio diamo maggior priorità al canale 0) oppure una priorità dinamica con politica Round-Robin. Dunque, le interruzioni provenienti dalle 2 PIA del nodo A non verranno più inoltrate direttamente al PROCESSORE, ma verranno indirizzate al DMA che si occupa della gestione delle interruzioni e della loro priorità, mentre solo quest'ultimo si collegherà alle linee di interruzione del PIC, che a sua volta indirizza una SOLA interruzione al PROCESSORE.

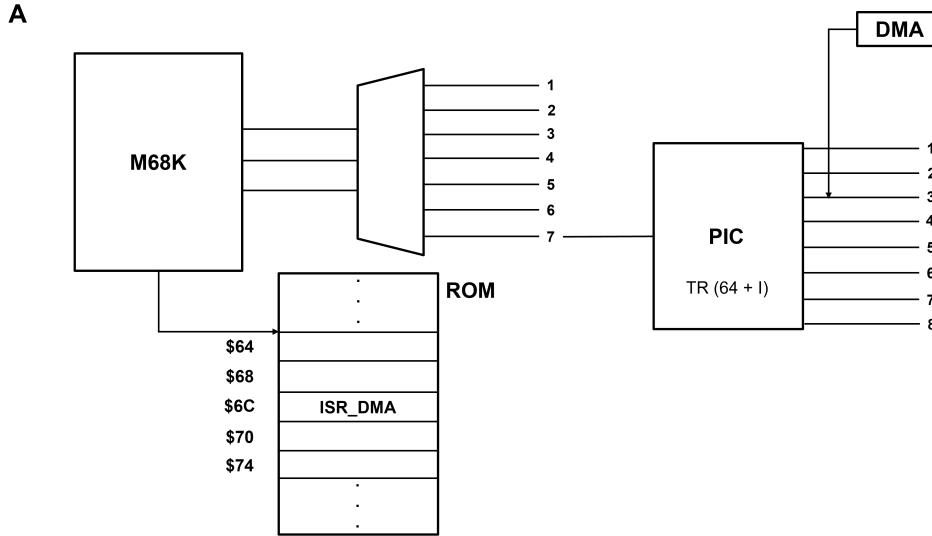


Figura 16: Livello delle interruzioni con PIC del nodo A

### 3.2 Protocolli

La logica del Driver a questo punto cambia in quanto c'è la necessità di configurare opportunamente l'interconnessione tra DMA e PERIFERICA. La configurazione sarà dunque settare prima il modo del DMA e poi quello delle rispettive periferiche. Il DMA deve fare una richiesta al processore per avvisare che vuole inserire l'indirizzo sul bus. Il processore deve fare una risposta, la quale consente di avviare il protocollo per andare a scrivere direttamente in memoria tramite il DMA. Tale protocollo è meglio illustrato nella Figura 17.

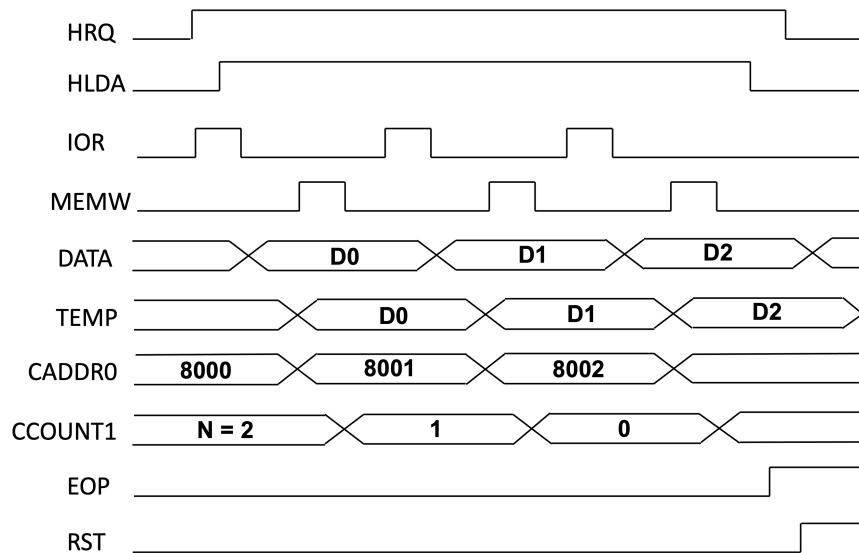


Figura 17: Protocollo DMA - DEV to MEM

### 3.3 Nuova logica del driver

La possibile configurazione in Assembly del DMA (PIA - DMA) e la relativa ISR di fine trasferimento, sarà la seguente:

```
...
    ORG $8200
MAIN    JSR DVAIN      *inizializzo le due PIA

*INIT_DMA
    MOVE.W #DMA, A0
    MOVE.W #MSG, CADDR0(A0)
    MOVE.B #NBYTES, CCOUNT1(A0)
    MOVE.B #$88, MODE0(A0)  *il bit 7 alto (modo block) e bit
                           *3 alto direzione interfaccia to mem
    MOVE.B #$80, CNTRL(A0)
    MOVE.B #$08, REQUEST(A0)

LOOP JMP LOOP

...
*ISR FINE TRASFERIMENTO DMA
    ORG $8700
INT7    NOP
        RTE

END     MAIN
```

Il DMA, dunque, se settato in **mode block**, trasferisce blocchi composti da N caratteri, a partire da una locazione di memoria fornita dall'indirizzo base. Una volta trasferiti questi N caratteri, il DMA genererà una SOLA interruzione per dire al processore che ha concluso l'operazione. Dunque, questo è il possibile vantaggio che si ha se si progetta l'architettura considerando anche il DMA in quanto se avessi, ad esempio, N=200 caratteri vuol dire che con la PIA dovrei interrompere per 200 volte la CPU, mentre con il DMA avrà un solo interrupt utile per occupare il bus e inviare il blocco dati in memoria.

Oltre, al DMA, occorre configurare ovviamente anche il PIC, al quale arriverà l'interrupt proveniente dal DMA per la fine del trasferimento, il codice di configurazione Assembly è il seguente:

```
...
PIA1DA EQU      $2004
PIA1CA EQU      $2005
PIA2DA EQU      $2006
PIA2CA EQU      $2007
PIC      EQU      $202C

...
ORG $8200
MAIN    JSR      INIT_PIA

*INIT_PIC
```

```

MOVEA.W #PIC ,A0
MOVE.B #$40 ,1(A0)           *VETTORE DI BASE DELLE INTERRUZIONI =64
                                *IN TR
MOVE.B #$10 ,( A0)           *AEOI=1
MOVE.B #$00 ,1(A0)           *ABILITO TUTTE LE INTERRUZIONI

MOVE.W SR,D0
ANDI.W #$D8FF ,D0
MOVE.W D0,SR

LOOP    JMP     LOOP

*ISR LINEA 1
        ORG      $8600
INT1    NOP
        RTE

*ISR LINEA 2
        ORG      $8700
INT2    NOP
        RTE

*ISR LINEA 3
        ORG      $8800
INT3    ...
        ISR_DMA
        ...
        RTE
        ...

END     MAIN

```

In definitiva, il PIC permette di gestire le interruzioni delle periferiche in maniera più flessibile rispetto al processore, inoltre, è possibile gestire in maniera più intelligente i conflitti che vi possono essere tra due periferiche usando gli 8 livelli di priorità del PIC stesso, che possono diventare 64 se collegiamo un ulteriore PIC in cascata. Tuttavia, il sistema con DMA e PIC appena illustrato non ha molto senso perché le 2 PIA sono collegate ai canali del DMA e quindi i conflitti sulle interruzioni sono gestite direttamente da quest'ultimo componente e quindi non ci sono conflitti che il PIC deve gestire, perché riceverà sempre una sola interruzione proveniente dal DMA.