



UNIVERSITA' DEGLI STUDI DI
NAPOLI FEDERICO II

Scuola Politecnica e delle Scienze di Base
Corso di Laurea Magistrale in Ingegneria Informatica

ELABORATO CSD

Anno Accademico 2022/2023

Candidato
PAOLO RUSSO
matr.M63001426

Contents

1	Specifiche di Progetto	1
2	Soluzione	5
2.1	Architettura complessiva	6
2.1.1	Collegamento Pia	6
2.2	Protocolli	7
2.3	Mappa della memoria	8
2.4	Descrizione di alto livello delle condizioni di funziona- mento	9
2.5	Descrizione altro livello del programma implementato .	10
2.5.1	PSEUDO_CODICE IRQB NODO A	10
2.5.2	PSEUDO_CODICE IRQC NODO A	11
2.6	Implementazione Assembly	12
2.6.1	IRQB NODO A	12
2.6.2	IRQC NODO A	15
2.7	Risultati Simulazione	19
2.8	Risposta Aggiuntiva PIC	20
2.8.1	Architettura nodo A con PIC	20
2.8.2	Possibili Approcci	20

2.9	Risposta Aggiuntiva DMA	21
2.9.1	Architettura nodo A con DMA	21
2.9.2	Possibili Approcci	22

Chapter 1

Specifiche di Progetto

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 effettuando K iterazioni (con $K > 2$ a scelta dello studente), in ciascuna delle quali A deve ricevere globalmente 2 messaggi di N caratteri da B e 1 messaggio di N caratteri da C (con $N > 2$ a scelta dello studente). I messaggi da B e da C possono essere ricevuti in un ordine qualsiasi ma non deve essere mai possibile ricevere caratteri appartenenti a messaggi diversi intervallati tra di loro. In altre parole, detto $msgB_i$ un generico messaggio completo ricevuto da B e $msgC_j$ un generico messaggio completo ricevuto da C, in ogni iterazione si possono avere le seguenti situazioni:

$$< msgB_1 msgB_2 msgC_1 > (s1)$$

$$< msgC_1 msgB_1 msgB_2 > (s2)$$

$$< msgB_1 msgC_1 msgB_2 > (s3)$$

Esempio: Un esempio con $K = 3; N = 3$ di funzionamento del sistema è il seguente:

- *Iter1*: $msgB_1(1)msgB_1(2)msgB_1(3) msgC_1(1)msgC_1(2)msgC_1(3)$
 $msgB_2(1)msgB_2(2)msgB_2(3) (s3)$
- *Iter2*: $msgC_1(1)msgC_1(2)msgC_1(3) msgB_1(1)msgB_1(2)msgB_1(3)$
 $msgB_2(1)msgB_2(2)msgB_2(3) (s2)$
- *Iter3*: $msgC_1(1)msgC_1(2)msgC_1(3) msgB_1(1)msgB_1(2)msgB_1(3)$
 $msgB_2(1)msgB_2(2)msgB_2(3) (s2)$

Si progetti e implementi l'unità A specificando:

1. Architettura complessiva: rappresentazione grafica schematica dell'architettura complessiva del sistema, in termini dei componenti di ciascuna unità (CPU, memoria, bus, dispositivi) e delle relative interconnessioni, in cui siano evidenziati i principali collegamenti e le linee di interruzione previste.
2. Protocolli: diagrammi temporali che rappresentino i principali protocolli di comunicazione utilizzati fra i dispositivi (ad es. i protocolli utilizzati per la scrittura e/o la lettura su/da periferica parallela).
3. Mappa della memoria: rappresentazione grafica schematica del contenuto della memoria RAM e ROM con riferimento alle aree

dati e codice del programma implementato e al vettore delle eccezioni (solo per la specifica unità richiesta).

4. Descrizione di alto livello delle condizioni di funzionamento considerate e dei meccanismi usati per garantire lo svolgimento della logica prevista dall'esercizio (es. è stato usato un flag che...); descrizione di alto livello dei problemi ravvisati di conflitto sui dati e/o di gestione di possibili “sovrapposizioni” di messaggi dovute alla diversa velocità di elaborazione dei dispositivi coinvolti) e delle principali soluzioni scelte (es. mutua esclusione con istruzione TAS, disattivazione selettiva interruzioni periferiche, ecc.).
5. Descrizione di alto livello del programma implementato: descrizione, mediante diagramma a blocchi o pseudocodice o automa, dei principali passi effettuati in ciascuno dei moduli software che compongono il programma (si richiede cioè un diagramma separato per il “main” e per ciascuna ISR prevista).
6. Implementazione: codice Assembly Motorola 68000 per il sistema progettato. Gli studenti sono invitati a inserire commenti nel codice almeno nelle parti salienti (ad esempio, nella configurazione delle periferiche e nell'utilizzo di variabili globali) per favorire una migliore leggibilità e comprensione dell'elaborato.

Dopo aver sviluppato l'intero progetto, si illustri come cambiereb-

bero l'architettura complessiva e la logica del driver se venisse inserito un PIC. Opzionalmente, si discuta cosa accade inserendo un DMA (lo studente scelga la configurazione più opportuna) specificando le eventuali modifiche necessarie alla logica del programma.

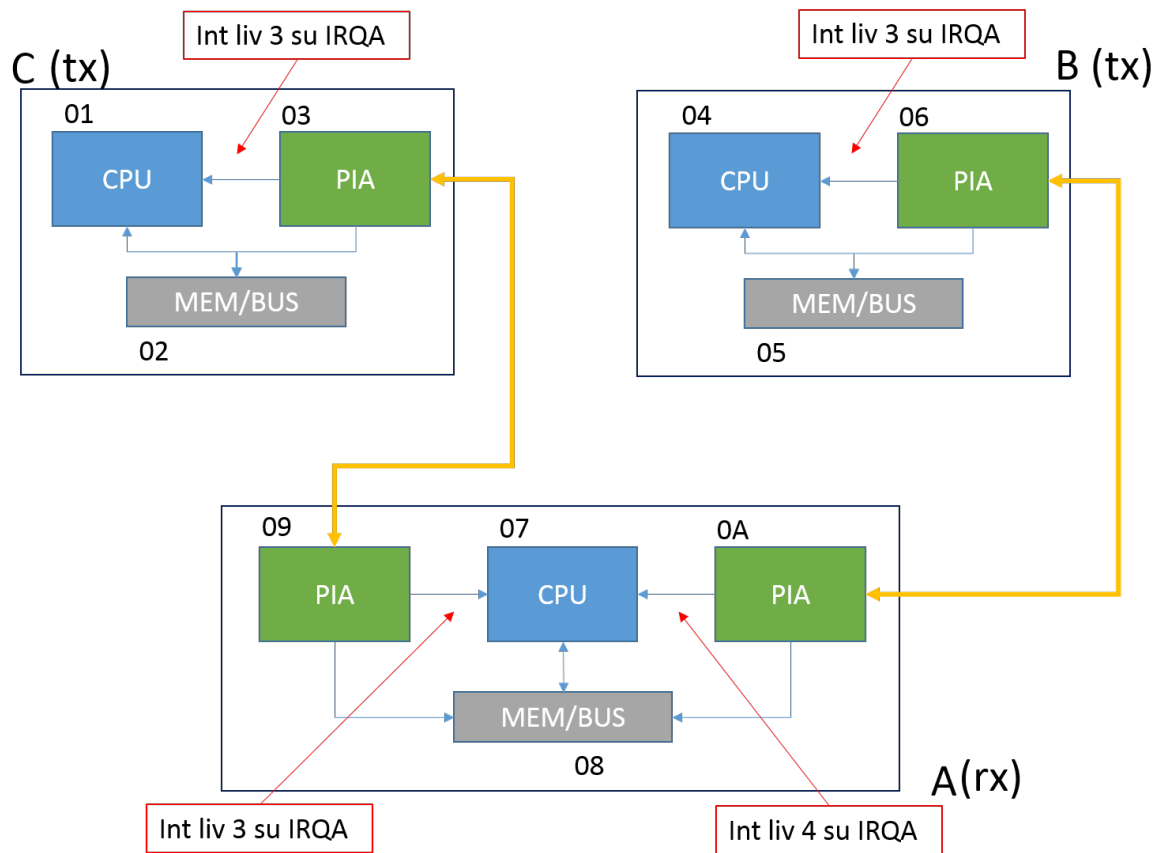
Nota: per rispondere alla domanda su PIC/DMA non è richiesta l'implementazione completa di un nuovo programma, ma lo studente dovrà indicare schematicamente le principali modifiche necessarie al codice assembly già prodotto (è preferibile a tale scopo indicare a parte gli stralci di codice da inserire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

Chapter 2

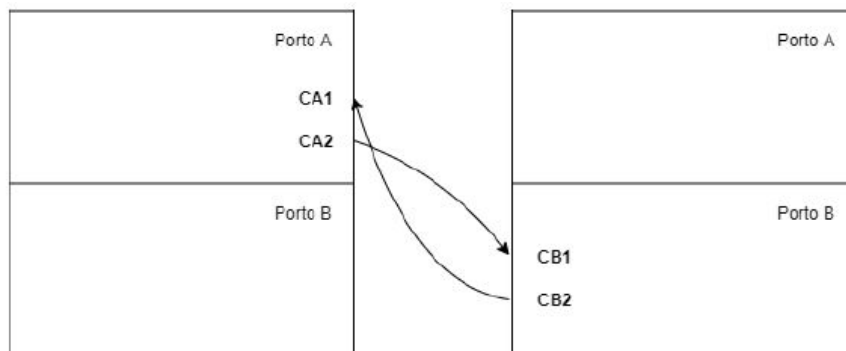
Soluzione

Andiamo a svolgere tutti i punti necessari, che partano dalla scelta del modello di programmazione, i protocolli delle rispettive periferiche in questo caso la PIA, la mappatura della memoria RAM e ROM con riferimento alle aree dati e codice del programma implementato e al vettore delle eccezioni fino ad arrivare allo sviluppo software.

2.1 Architettura complessiva



2.1.1 Collegamento Pia



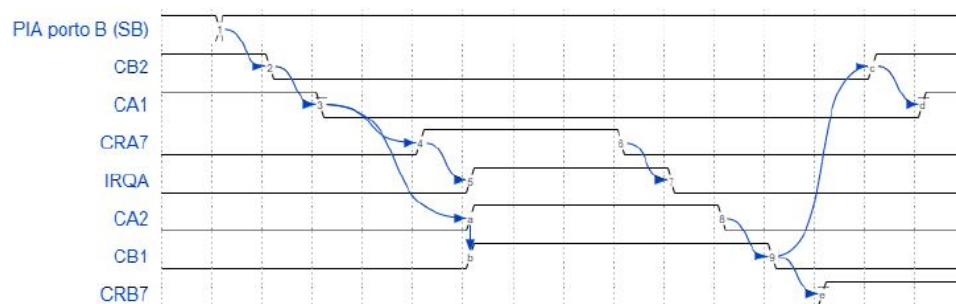
Per quanto riguarda le PIA, quella del SistemaA è configurata in ricezione sul porto A, mentre la PIA del SistemaB è configurata in

trasmissione sul porto B. IL protocollo di comunicazione, tra le due PIA, funziona nel seguente modo :

- In uscita dal porto B della PIA del SistemaB abbiamo il segnale CB2, che va in entrata al segnale CA1, che comunica l'interruzione al SistemaA;
- In uscita dal porto A della PIA del SistemaA abbiamo il segnale CA2, che va in entrata al segnale CB1, che comunica l'interruzione al SistemaB.

2.2 Protocolli

Nella figura viene rappresentata la comunicazione tra le due PIA: nel momento in cui il porto B della PIA comincia la trasmissione abbassa CB2, che a sua volta è collegato al segnale CA1. Quando quest'ultimo si abbassa, si alza CRA7 (bit collegato all'interruzione del Porto A), scatenando di conseguenza l'alzarsi del segnale IRQA, CA2 e CB1. Nel momento in cui l'ISR effettua la lettura, invece ,si abbassano i segnali IRQA, CA2 e CB1, causando l'alzarsi del segnale CRB7, il quale si abbasserà soltanto con una lettura fittizia.



2.3 Mappa della memoria

- Mappatura PIA_C

PIABD_C	\$2004
PIABC_C	\$2005

- Mappatura PIA_D

PIABD_B	\$2008
PIABC_B	\$2009

- Mappatura vettore delle eccezioni

La modalità di riconoscimento dell'interruzione utilizzata è quella autovettorizzata, dove il dispositivo che interrompe richiede una gestione automatica dell'interruzioni. Per calcolare l'indirizzo, dunque, dobbiamo *fare* $(24 + IPL) * 4$.

IRQB	\$6C
IRQC	\$70

- Mappatura aree dati e codice del programma

AREA DATI	\$8000
AREA MAIN	\$8300
IRQ_B	\$8700
IRQ_C	\$8900

2.4 Descrizione di alto livello delle condizioni di funzionamento

1. La PIA_B ha già ricevuto un messaggio intero e due caratteri del secondo messaggio. Arriva l'interruzione di C che entra nell'area critica. Poi, arriva l'interruzione di B che provando ad entrare nell'area critica si sospende. C capisce che non è il suo turno e si sospende e poi risveglia B che finisce di leggere l'intero messaggio fa le sue operazioni e poi risveglia C.

B: B2 B2 B2 B2

C:.....#.....C C C C C

2. La PIA_C ha già ricevuto due caratteri del messaggio arriva l'interruzione di C che entra nell'area critica, esce e sta per fare Rx_C. Arriva l'interruzione di B che entra nell'area critica ma capisce che non è il suo turno e si sospende C completa il messaggio, fa le sue operazioni e risveglia B.

B: #.....B

C: C C C C C

3. C ha già letto il suo messaggio arriva un'interruzione di C che capisce che non è ancora il suo turno e si sospende.

B:

C: C1 C1 C1 C1 C1 SOS_C

2.5 Descrizione altro livello del programma implementato

2.5.1 PSEUDO_CODICE IRQB NODO A

```

IRQ_B{
  tas(LOCK):
    if(possesso è di B or possesso di nessuno):
      assegno il possesso a B
      LOCK= 0
      Rx_B
      incremento contatore caratteri ricevuti di B

    if(ho ricevuto N caratteri):
      azzero il contatore dei caratteri ricevuti di B
      incremento contatore messaggi ricevuti di B

    if(ho ricevuto 2 messaggi):
      azzero il contatore dei caratteri ricevuti di B
      alzo FLG_B per segnalare l'avvenuta lettura del secondo messagg:
      imposto possesso a 0
      if(FLG_C è alto):
        incremento il contatore delle iterazioni K
        if(ho effettuato k iterazioni):
          disabilito le periferiche
        else:
          azzero FLG_B
          azzero FLG_C
          if(C è sospeso):
            risveglio C
      else:
        if(C è sospeso):
          risveglio C
      else:
        imposto possesso a 0
        if(C è sospeso):
          risveglio C
    else:
      fine_b
  else:
    LOCK=0
    sospendi B
  else:
    sospendo B
  fine_B
}

```

2.5.2 PSEUDO_CODICE IRQC NODO A

```

IRQ_C{
  tas(LOCK):
    if(posesso è di C o possesso non è assegnato):
      imposto il possesso a C
      LOCK=0
      Rx_C
      incremento il contatore die caratteri ricevuti di c
      if(ho ricevuto N caratteri):
        azzero il contatore dei caratteri ricevuti di C
        alzo il FLG_C per segnalare la lettura dell'intero messaggio
        if(FLG_B è alto):
          incremento il contatore delle interazioni k
          if(ho effettuato k iterazioni):
            disabilitop le periferiche
          else:
            azzero FLG_C
            azzero FLB_B
            imposto possesso a 0
            if(B è sospeso):
              risveglio B
          else:
            imposto il possesso a 0
            if(B è sospeso):
              risveglio B
        else:
          fine_c
      else:
        LOCK=0
        sospendo C
        if(B è sospeso):
          risveglio B
    else:
      sospendo C
      if(B è sospeso):
        risveglio B
  fine_C
}

```

2.6 Implementazione Assembly

2.6.1 IRQB NODO A

	ORG	\$8700
IRQ_B	MOVEM.L	D0-D1/A0-A1,-(A7)
IF_B_1	TAS	LOCK
	BNE	ELSE_B_1
	MOVE.B	POS,D0
IF_B_2	CMP.B	#1,D0
	BEQ	SKIPB
	CMP.B	#0,D0
	BNE	ELSE_B_2
SKIPB	MOVE.B	#1,POS
	MOVE.B	#0,LOCK
RX_B	MOVEA.L	#PIABD_B,A0
	MOVEA.L	#MSG_B,A1
	MOVE.B	CONT_C_B,D0
	MOVE.B	(A0),(A1,D0)
	ADD.B	#1,D0
	MOVE.B	D0,CONT_C_B
	MOVE.B	N,D1
IF_B_3	CMP	D0,D1
	BNE	FINE_B
	MOVE.B	#0,CONT_C_B
	MOVE.B	CONT_M_B,D0
	ADD.B	#1,D0
	MOVE.B	D0,CONT_M_B
	MOVE.B	M_B,D1

IF_B_4	CMP	D0,D1
	BNE	ELSE_B_4
	MOVE.B	#0,CONT_M_B
	MOVE.B	#1,FLG_B
	MOVE.B	#0,POS
	MOVE.B	FLG_C,D0
IF_B_5	CMP.B	#1,D0
	BNE	RISVEGLIO_C
	MOVE.B	CONT_K,D0
	ADD.B	#1,D0
	MOVE.B	D0,CONT_K
	MOVE.B	K,D1
IF_B_6	CMP.B	D0,D1
	BNE	ELSE_B_6
	MOVE.B	#0,PIABC_B
	MOVE.B	#0,PIABC_C
	BRA	FINE_B
ELSE_B_2	MOVE.B	#0,LOCK
ELSE_B_1	MOVE.B	#1,SOS_B
	BRA	FINE_B
ELSE_B_4	MOVE.B	#0,POS
	BRA	RISVEGLIO_C
ELSE_B_6	MOVE.B	#0,FLG_B
	MOVE.B	#0,FLG_C
RISVEGLIO_C	MOVE.B	SOS_C,D0

IF_B_7	CMP.B	#1,D0
	BNE	FINE_B
	MOVE.B	#2,POS
	MOVE.B	#0,SOS_C
	MOVE.B	CONT_C_C,D0
	MOVEA.L	#PIABD_C,A0
	MOVEA.L	#MSG_C,A1
	MOVE.B	(A0),(A1,D0)
	ADD.B	#1,D0
	MOVE.B	D0,CONT_C_C
FINE_B	MOVEM.L	(A7)+,A1-A0/D1-D0
	RTE	

2.6.2 IRQC NODO A

	ORG	\$8900
IRQ_C	MOVEM.L	D0-D2/A0-A1,-(A7)
IF_C_1	TAS	LOCK
	BNE	ELSE_C_2
	MOVE.B	FLG_C,D0
	CMP.B	#0,D0
	BNE	ELSE_C_2
	MOVE.B	POS,D0
	CMP.B	#2,D0
	BEQ	SKIPC
IF_C_2	CMP.B	#0,D0
	BNE	ELSE_C_2
	MOVE.B	#2,POS
SKIPC	MOVE.B	#0,LOCK
RX_C	MOVEA.L	#PIABD_C,A0
	MOVEA.L	#MSG_C,A1
	MOVE.B	CONT_C_C,D0
	MOVE.B	(A0),(A1,D0)
	ADD.B	#1,D0
	MOVE.B	D0,CONT_C_C
	MOVE.B	N,D1
IF_C_3	CMP.B	D0,D1
	BNE	FINE_C
	MOVE.B	#0,CONT_C_C
	MOVE.B	#1,FLG_C
	MOVE.B	FLG_B,D0

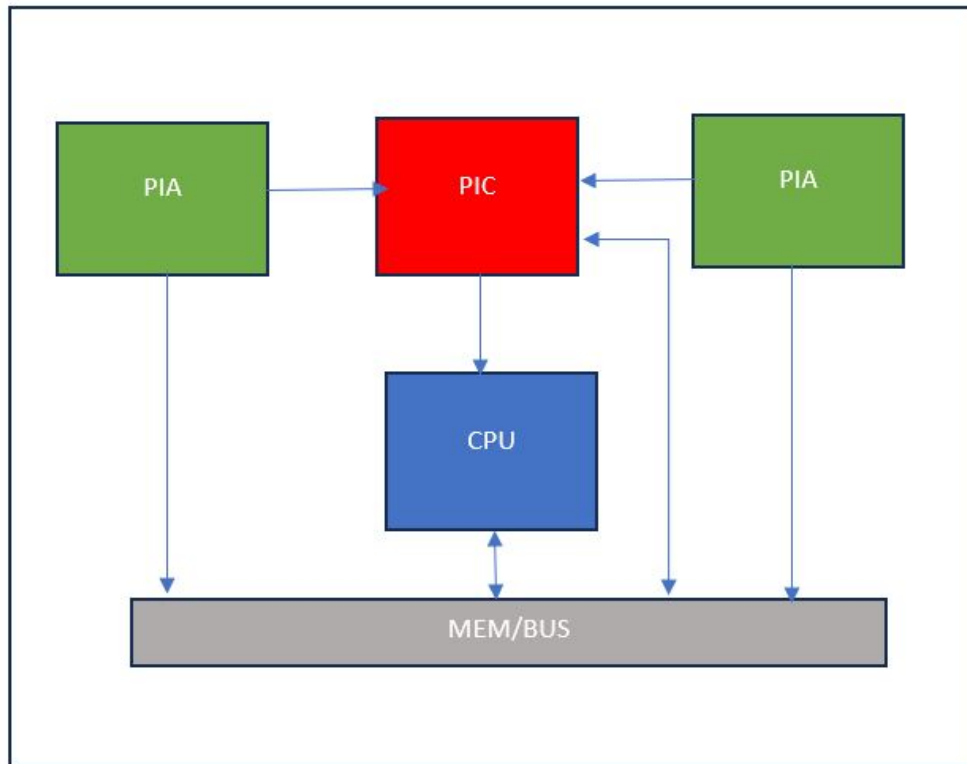
IF_C_4	CMP.B	#1,D0
	BNE	ELSE_C_4
	MOVE.B	CONT_K,D0
	ADD.B	#1,D0
	MOVE.B	D0,CONT_K
	MOVE.B	K,D1
IF_C_5	CMP.B	D0,D1
	BNE	ELSE_C_5
	MOVE.B	#0,PIABC_C
	MOVE.B	#0,PIABC_B
	BRA	FINE_C
ELSE_C_2	MOVE.B	#0,LOCK
	MOVE.B	#1,SOS_C
RIS_B_TOTALE	MOVE.B	SOS_B,D0
	CMP.B	#1,D0
	BNE	FINE_C
	MOVE.B	#1,POS
	MOVE.B	#0,SOS_B
	MOVE.B	CONT_C_B,D0
	MOVE.B	CONT_C_B,D1
	ADD.B	#1,D1
	MOVE.B	D1,CONT_C_B
	MOVE.B	N,D2
IF_C_6	CMP.B	D1,D2
	BNE	ELSE_C_6
	MOVE.B	#0,CONT_C_B
	MOVE.B	CONT_M_B,D2
	ADD.B	#1,D2

	MOVE.B	D2,CONT_M_B
	MOVE.B	M_B,D1
IF_C_7	CMP.B	D1,D2
	BNE	ELSE_C_8
	MOVE.B	#1,FLG_B
	MOVE.B	#0,CONT_M_B
	MOVE.B	#0,POS
	MOVE.B	FLG_C,D1
IF_C_8	CMP.B	#1,D1
	BNE	ELSE_C_8
	MOVE.B	CONT_K,D1
	ADD.B	#1,D1
	MOVE.B	D1,CONT_K
	MOVE.B	K,D2
IF_C_9	CMP.B	D1,D2
	BNE	ELSE_C_9
	MOVEA.L	#PIABD_B,A0
	MOVEA.L	#MSG_B,A1
	MOVE.B	(A0),(A1,D0)
	MOVE.B	#0,PIABC_C
	MOVE.B	#0,PIABC_B
	BRA	FINE_C
ELSE_C_9	MOVE.B	#0,FLG_B
	MOVE.B	#0,FLG_C
ELSE_C_8	MOVE.B	#2,POS
	MOVEA.L	#PIABD_B,A0
	MOVEA.L	#MSG_B,A1
	MOVE.B	(A0),(A1,D0)
	JMP	RX_C

ELSE_C_6	MOVEA.L	#PIABD_B,A0
	MOVEA.L	#MSG_B,A1
	MOVE.B	(A0),(A1,D0)
	BRA	FINE_C
ELSE_C_4	MOVE.B	#0,POS
	BRA	RIS_B
ELSE_C_5	MOVE.B	#0,FLG_C
	MOVE.B	#0,FLG_B
	MOVE.B	#0,POS
RIS_B	MOVE.B	SOS_B,D0
	CMP.B	#1,D0
	BNE	FINE_C
	MOVE.B	#1,POS
	MOVE.B	#0,SOS_B
	MOVE.B	CONT_C_B,D0
	MOVE.B	CONT_C_B,D1
	ADD.B	#1,D1
	MOVE.B	D1,CONT_C_B
	MOVEA.L	#PIABD_B,A0
	MOVEA.L	#MSG_B,A1
	MOVE.B	(A0),(A1,D0)
FINE_C	MOVEM.L	(A7)+,A1-A0/D2-D0
	RTE	

2.8 Risposta Aggiuntiva PIC

2.8.1 Architettura nodo A con PIC



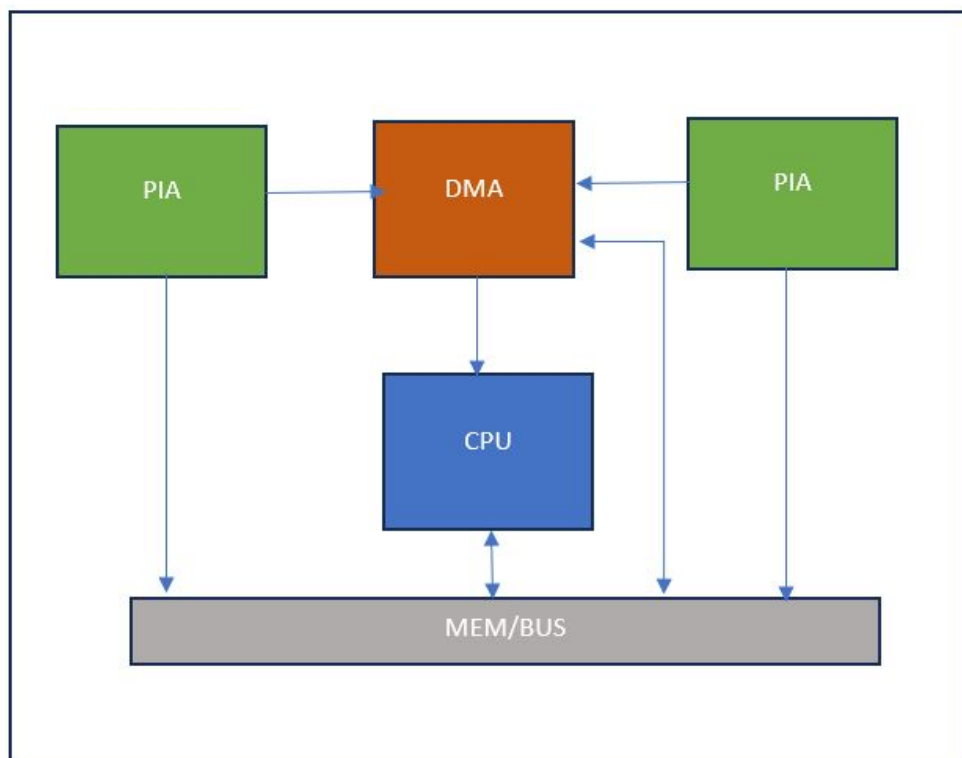
2.8.2 Possibili Approcci

Il PIC è un device che estende le funzionalità di gestore delle interruzioni via hardware, permettendo di implementare fino ad 8 livelli di priorità. Nella pratica, dopo aver inizializzato gli indirizzi delle mie periferiche, vado a settare nel registro TR il valore del vettore base che rappresenta le ISR definite dall'utente. Dopodichè, setto il bit AEIOI=1, per il reset automatico, agendo sul registro CTRL. Infine, imposto il registro IMR, per poter abilitare le linee d'interruzione.

Dopo, aver definito le rispettive ISR, bisogna mapparle, partendo dalla locazione $64 * 4 = \$100$. Quindi, a partire da questa locazione base mapperò la IRQB mentre l'altra verrà mappata nella locazione $\$104$. Poichè ASIM è un ambiente di simulazione, per osservare il comportamento del PIC, avremmo bisogno di un timer, rappresentato dal dispositivo 1T04INTGEN, il quale si interfaccia direttamente con il PIC e genera un'interruzione, ogni qualvolta scade il timer prefissato, generando un ISR.

2.9 Risposta Aggiuntiva DMA

2.9.1 Architettura nodo A con DMA



2.9.2 Possibili Approcci

Potrei optare tre eventuali soluzioni.

1. Uso i canali in maniera mutuamente esclusiva, in cui ogni volta che inizia B, disabilito C e viceversa.
2. Imponendo un ordine, posso abilitare B dopo aver terminato lo disabilito, e applico lo stesso meccanismo con C.
3. Se non faccio nessuna ipotesi, non posso usare il DMA, poichè i caratteri arriveranno in maniera casuale.