Prova 22/05/2025

Rocco Lo Russo
roc.lorusso@studenti.unina.it

Università di Napoli Federico II - DIETI — 15/07/2025

Introduzione

In questo documento verrà sviluppata la prova intercorso del 22/05/2025, esponendo analiticamente i seguenti punti: **mappa della memoria**, **pseudocodice** e implementazione in **ASIM**.

1 Traccia

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. I messaggi hanno un primo carattere identificativo che può essere pari a 0 a un valore diverso da 0. Il sistema opera in due fasi successive come descritto di seguito:

- Fase 1: A riceve K messaggi di N caratteri da B e da C in modo alternato. In ordine non prefissato, quindi si parte da B o da C, e **non ci sono sovrapposizioni tra i messaggi ricevuti da B o da C**;
- Fase 2: Al termine della fase 1, il nodo A continua nella stessa modalità alternata e termina la ricezione dei messaggi se due messaggi ricevuti (dalle due diverse periferiche) hanno il carattere identificativo del messaggio pari a 0.

1.1 Analisi della traccia

Dalla traccia emerge che la comunicazione debba essere gestita in modo tale da garantire la *non sovrapposizione* dei messaggi ricevuti da B e da C. Questo implica che verrà utilizzato un unico buffer di ricezione in cui verranno conservati i messaggi ricevuti, e che una periferica non potrà mandare un nuovo carattere se l'altra non avrà finito la trasmissione di un intero messaggio. La ricezione dei messaggi in modo alternato pone il vincolo, nella codifica della ISR, di considerare casuale la provenienza del primo messaggio. Ci sono due modi di interpretare questo passo della traccia:

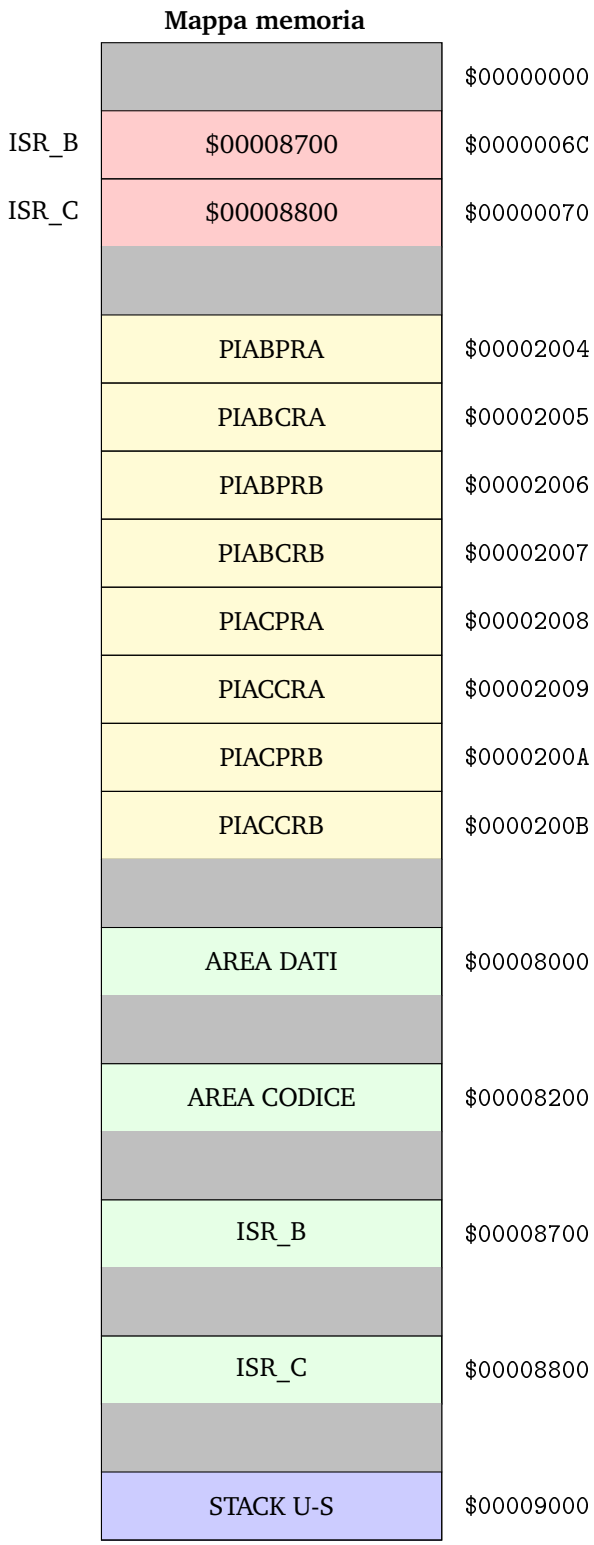
- Il primo messaggio può arrivare da qualsiasi periferica, e una volta arrivato questo stabilisce l'ordine di arrivo di tutti gli altri K-1 messaggi;
- I messaggi vengono ricevuti in modo alternato ma a coppie, ovvero **per ogni coppia** il primo messaggio può provenire da qualsiasi periferica, ma il secondo deve provenire necessariamente dall'altra.

La soluzione presentata più avanti si basa sulla seconda interpretazione. Sotto questi vincoli, i conflitti da gestire sono:

- Accesso in mutua esclusione in scrittura alla risorsa rappresentata dal nodo A, in modo da garantire la non sovrapposizione dei messaggi;
- Regolare l'accesso alla risorsa in modo che sia alternato, curando i casi in cui le ISR pongano una periferica in stato di attesa.

2 Mappa della memoria

In questa sezione verrà presentata una mappa della memoria del nodo A in accordo a quanto specificato nel file .cfg utilizzato nella simulazione e in accordo alla memoria caricata (*file rom.mem*).



3 Implementazione

In questa sezione verrà presentato il codice assembly per Motorola68000 e lo pseudocodice usato come riferimento per l'implementazione.

3.1 Variabili

Descrizione delle variabili utilizzate:

`fine`	Intero che può assumere i valori 0 (il nodo A è in ricezione) o 1 (il nodo A ha terminato la ricezione).
`lock`	Intero che può assumere i valori 0 o 1, viene testato dall'istruzione atomica TAS per garantire l'accesso mutualmente esclusivo alla sezione critica.
`possesso`	Intero che può assumere i valori -1 (`is_free`), 0 (`is_reading_b`) o 1 (`is_reading_c`).
`buff`	Puntatore alla prima locazione di un vettore di dimensione K*N di caratteri; serve per accedere alla memoria del nodo A.
`curr`	Intero da 0 a N che tiene conto dei caratteri ricevuti.
`tot`	Intero da 0 a K*N per l'accesso indirizzato al vettore dei caratteri.
`msg`	Intero da 0 a K che conta i messaggi ricevuti.
`end_b,(end_c)`	Intero: 0 se il messaggio da b (c) non è terminato, 1 altrimenti.
`fase2`	Intero: 0 (fase 1), 1 (fase 2).
`cond_b,(cond_c)`	Intero: 0 se il primo carattere ricevuto da b (c) non è 0, 1 altrimenti.
`b_sus,(c_sus)`	Intero: 0 se b (c) non è bloccato, 1 se è in attesa che venga letto il carattere.
`idx`	Variabile temporanea per memorizzare un valore della variabile `tot`.

3.2 Pseudocodice

Assumiamo che ISR_B e ISR_C siano speculari, e che ISR_B sia più prioritaria di ISR_C : se il nodo A riceve un messaggio da C durante l'esecuzione di ISR_B, ISR_C prelaziona ISR_B. Come anticipato nella sezione 1, l'accesso alla sezione critica in cui si controlla ed eventualmente modifica la variabile *possesso* avverrà in mutua esclusione.

```
1  # define is_reading_b 0
2  # define is_reading_c 1
3  # define is_free -1
4
5  void isr_b(){
6      if(!fine){
7          if(TAS(lock)){
8              if(possesso !=c and !end_b){
9                  possesso = is_reading_b;
10             }
11             lock = 0;
12         }else{
13             RTE;
14         }
15
16         switch (possesso){
17             case is_reading_b:
18                 buff[tot] = PIABPRA;
19                 if(curr == 0 && fase2 && buff[tot] == 0){
20                     cond_b = 1;
21                 }
22                 tot++;
23                 curr++;
24                 if(curr == N){
25                     curr = 0;
26                     msg++;
27                     end_b = 1;
28                     possesso = is_free;
29                     if(end_c){
30                         end_b = 0;
31                         end_c = 0;
32                         if(msg==k){
33                             fase2=1;
34                         }
35                     }
36                     if(fase2 && (cond_b && cond_c)){
37                         fine = 1;
38                     }
39                     if(c_sus && !end_c){
40                         tot++;
41                         curr++;
42                         buff[tot-1] = PIACPRA;
43                         // se c interrompe qui, trovera' possesso=is_free
44                         if(buff[tot-1] == 0 && fase2){
45                             cond_c=1;
46                         }
47                         possesso = is_reading_c;
48                     }
49                 }
50              case is_free:
51                 if(c_sus && !end_c){
52                     tot++;
53                     curr++;
```

```
54                      buff[tot-1] = PIACPRA;
55                      if(buff[tot-1] == 0 && fase2){
56                          cond_c=1;
57                      }
58                      possesso = is_reading_c;
59                  }
60              case is_reading_c{
61                  if(c_sus){
62                  // L'unica assunzione possibile e' che
63                  // questo non sia il primo carattere di b
64                      idx = tot;
65                      tot++;
66                      curr++;
67                      possesso = is_reading_c;
68                      if(curr == N){
69                          curr = 0;
70                          msg++;
71                          end_c = 1;
72                          possesso = is_free;
73                          if (end_b){
74                              end_b = 0;
75                              end_c = 0;
76                              if(msg == k){
77                                  fase2=1;
78                              }
79                          }
80                          if(fase2 && (cond_b && cond_c)){
81                              fine = 1;
82                          }
83                          if(!end_b){
84                              buff[tot]=PIABPRA;
85                              tot++;
86                              curr++;
87                              if(fase2 && buff[tot-1]==0){
88                                  cond_b = 1;
89                              }
90                              possesso = is_reading_b;
91                          }
92                      }
93                      buff[idx]=PIACPRA;
94                  }
95              }
96          }
97          RTE;
98      }else{  // se fine == 1
99          RTE;
100     }
101 } // fine isr_b
```

3.3 Codice assembly MC68000

Di seguito viene esposta la codifica in assembly, basata sullo pseudocodice esposto nel paragrafo precedente, del programma eseguito dal nodo A e delle ISR relative alla ricezioni di caratteri sulla parallela proveniente da B e C. I programmi eseguiti dai nodi B e C consistono in semplici cicli di invio di messaggi tramite parallela.

```
1          ORG      $8000        * AREA DATI
2  FINE    DC.B     0
3  LOCK    DC.B     0
4  POSS    DC.B     -1
5  CURR    DC.B     0
6  TOT     DC.B     0
7  MSG     DC.B     0
8  END_B   DC.B     0
9  END_C   DC.B     0
10 FASE2   DC.B     0
11 COND_B  DC.B     0
12 COND_C  DC.B     0
13 BUFF    DS.B     18
14
15         ORG      $8200        * AREA CODICE
16 PIABPRA EQU      $2004
17 PIABCRA EQU      $2005
18 PIACPRA EQU      $2008
19 PIACCRA EQU      $2009
20 N       EQU      3
21 K       EQU      6
22
23 MAIN    JSR      PIAINIT
24         MOVE.W   SR,D0
25         ANDI.W   #$D8FF,D0
26         MOVE.W   D0,SR
27 LOOP    JMP      LOOP
28
29
30 PIAINIT MOVE     #$00,PIABCRA
31         MOVE     #$00,PIABPRA
32         MOVE     #%00100101,PIABCRA
33         MOVE     #$00,PIACCRA
34         MOVE     #$00,PIACPRA
35         MOVE     #%00100101,PIACCRA
36         RTS
37
38         ORG      $8700
39 ISR_B   MOVEM.L  D0-D7/A0-A2,-(SP)
40         MOVE     FINE,D0
41         CMP      #$01,D0
42         BEQ      RETURN
43         TAS      LOCK
44         BNE      RETURN
45         MOVE     POSS,D0
46         CMP      #1,D0
47         BEQ      CONT0
48         MOVE     END_B,D0
49         CMP      #1,D0
50         BEQ      CONT0
51         MOVE     #0,POSS
52 CONT0   MOVE     #0,LOCK
53         MOVE     POSS,D0
```

```
54          CMP       #1,D0
55          BEQ       CASE1
56          CMP       #-1,D0
57          BEQ       CASE2
58          MOVEA.L   #PIABPRA,A0       * is_reading_b
59          MOVEA.L   #BUFF,A1
60          MOVE      TOT,D0
61          MOVE      CURR,D1
62          MOVE      (A0),(A1,D0)
63          MOVE      (A1,D0),D2
64          MOVE      FASE2,D3
65          CMP       #0,D2
66          BNE       CONT1
67          CMP       #1,D3
68          BNE       CONT1
69          CMP       #0,D1
70          BNE       CONT1
71          MOVE      #1,COND_B
72 CONT1    ADDQ      #1,D0
73          ADDQ      #1,D1
74          MOVE      D0,TOT
75          MOVE      D1,CURR
76          CMP       #N,D1
77          BNE       RETURN
78          MOVE      #0,CURR
79          MOVE      MSG,D0
80          ADDQ      #1,D0
81          MOVE      D0,MSG
82          MOVE      #1,END_B
83          MOVE      #-1,POSS
84          MOVE      END_C,D1
85          CMP       #1,D1
86          BNE       CONT2
87          MOVE      #0,END_B
88          MOVE      #0,END_C
89          CMP       K,D0
90          BNE       CONT2
91          MOVE      #1,FASE2
92 CONT2    MOVE      FASE2,D0
93          CMP       #1,D0
94          BNE       CONT3
95          MOVE      END_B,D0
96          CMP       #1,D0
97          BNE       CONT3
98          MOVE      END_C,D0
99          CMP       #1,D0
100         BNE       CONT3
101         MOVE      #1,FINE           * controllo c_sus
102 CONT3   MOVE      PIACCRA,D0
103         ANDI      #%10000000,D0
104         BEQ       RETURN
105         MOVE      END_C,D0
106         CMP       #1,D0
107         BNE       RETURN
108         MOVEA.L   #PIACPRA,A0
109         MOVEA.L   #BUFF,D0
110         MOVE      TOT,D0
111         MOVE      TOT,D7
112         MOVE      CURR,D1
113         ADDQ      #1,D0
114         ADDQ      #1,D1
115         MOVE      D0,TOT
```

```
116           MOVE     D1,CURR
117           MOVE     (A1),(A0,D7)
118           MOVE     (A0,D7),D0
119           MOVE     FASE2,D1
120           CMP      #1,D1
121           BNE      CONT4
122           CMP      #0,D0
123           BNE      CONT4
124           MOVE     #1,COND_C
125 CONT4     MOVE     #1,POSS
126           JMP      RETURN
127 CASE1     MOVE     PIACCRA,D0       * is_reading_c
128           ANDI     #%10000000,D0
129           BEQ      RETURN
130           MOVE     TOT,D7
131           MOVE     TOT,D0
132           MOVE     CURR,D1
133           ADDQ     #1,D0
134           ADDQ     #1,D1
135           MOVE     D0,TOT
136           MOVE     D1,CURR
137           MOVE     #1,POSS
138           CMP      N,D1
139           BNE      CONT5
140           MOVE     #0,CURR
141           MOVE     MSG,D0
142           ADDQ     #1,D0
143           MOVE     D0,MSG
144           MOVE     #1,END_C
145           MOVE     #-1,POSS
146           MOVE     END_B,D0
147           CMP      #1,D0
148           BNE      CONT6
149           MOVE     #0,END_B
150           MOVE     #0,END_C
151           MOVE     MSG,D0
152           CMP      K,D0
153           BNE      CONT6
154           MOVE     #1,FASE2
155 CONT6     MOVE     FASE2,D0
156           CMP      #1,D0
157           BNE      CONT7
158           MOVE     COND_B,D0
159           CMP      #1,D0
160           BNE      CONT7
161           MOVE     COND_C,D0
162           CMP      #1,D0
163           BNE      CONT7
164           MOVE     #1,FINE
165 CONT7     MOVE     END_B,D0
166           CMP      #1,D0
167           BNE      CONT5
168           MOVE     TOT,D0
169           MOVE     CURR,D1
170           MOVEA.L  #PIABDRA,A0
171           MOVEA.L  #BUFF,A1
172           MOVE     (A0),(A1,D0)
173           MOVE     (A1,D0),D2
174           ADDQ     #1,D0
175           ADDQ     #1,D1
176           MOVE     D0,TOT
177           MOVE     D1,CURR
```

```
178           MOVE      FASE2,D3
179           CMP       #1,D3
180           BNE       CONT8
181           CMP       #0,D2
182           BNE       CONT8
183           MOVE      #1,COND_B
184 CONT8     MOVE      #0,POSS
185 CONT5     MOVEA.L   #PIACPRA,A0
186           MOVEA.L   #BUFF,A1
187           MOVE      (A0),(A1,D7)
188           JMP       RETURN
189 CASE2     MOVE      PIACCRA,D0        * is_free
190           ANDI      #%10000000,D0
191           BEQ       RETURN
192           MOVE      END_C,D0
193           CMP       #1,D0
194           BEQ       RETURN
195           MOVE      TOT,D0
196           MOVE      CURR,D1
197           MOVEA.L   #PIACPRA,A0
198           MOVEA.L   #BUFF,A1
199           MOVE      D0,D7
200           ADDQ      #1,D0
201           ADDQ      #1,D1
202           MOVE      D0,TOT
203           MOVE      D1,TOT
204           MOVE      (A0),(A1,D7)
205           MOVE      (A1,D7),D2
206           MOVE      FASE2,D3
207           CMP       #1,D3
208           BNE       CONT9
209           CMP       #0,D2
210           BNE       CONT9
211           MOVE      #1,COND_C
212 CONT9     MOVE      #1,POSS
213           JMP       RETURN
214 RETURN    MOVEM.L   (SP)+,D0-D7/A0-A2
215           RTE
```

La ISR_C è stata omessa per motivi di spazio, essendo perfettamente speculare alla ISR_B.