Prova 22/05/2025

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

Università di Napoli Federico II - DIETI — 13/07/2025

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 l'altro deve provenire necessariamente dall'altra.

La soluzione presentata più avanti si basa sulla seconda ipotesi. 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

Todo: Implementa uno schema con la mappa della memoria dopo aver sistemato il file cfg ASIM.

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 LOCK 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`	Variabile intera 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  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){
```

```
                cond_b = 1;
            }
            tot++;
            curr++;
            if(curr == N){
                curr = 0;
                msg++;
                end_b = 1;
                possesso = is_free;
                if(end_c){
                    end_b = 0;
                    end_c = 0;
                    if(msg==k){
                        fase2=1;
                    }
                }
                if(fase2 && (cond_b && cond_c)){
                    fine = 1;
                }
                if(c_sus && !end_c){
                    tot++;
                    curr++;
                    buff[tot-1] = PIACPRA;
                    if(buff[tot-1] == 0){
                        cond_c=1;
                    }
                    possesso = is_reading_c;
                }
            }
         case is_free:
            if(c_sus && !end_c){
                tot++;
                curr++;
                buff[tot-1] = PIACPRA;
                if(buff[tot-1] == 0){
                    cond_c=1;
                }
                possesso = is_reading_c;
            }
        case is_reading_c{
            if(c_sus){
                idx = tot;
                tot++;
                curr++;
                if(curr == N){
                    curr = 0;
                    msg++;
                    end_c = 1;
                    possesso = is_free;
                    if (end_b){
                        end_b = 0;
                        end_c = 0;
                        msg++;
                        if(msg == k){
                            fase2=1;
                        }
                    }
                    if(fase2 && (cond_b && cond_c)){
                        fine = 1;
                    }
                    if(!end_b){
                        buff[tot]=PIABPRA;
```

```
82                                  tot++;
83                                  curr++;
84                                  if(fase2 && buff[tot-1]==0){
85                                      cond_b = 1;
86                                  }
87                                  possesso = is_reading_b;
88                              }
89                          }
90                          possesso = is_reading_c;
91                      }
92                  }
93              }
94              RTE;
95          }else{
96              RTE;
97          }
98  }
```