# Sistemi Operativi e Laboratorio, Prova del 4/4/2019 – versione B

Nome e cognome: _____ Matricola:_____ corso:____ aula: __fila:__ posto:___

**Esercizio 1 (4 punti)**
In un sistema che implementa i thread a livello del nucleo, dire in quale stato del processore (utente/supervisore o entrambi) possono essere eseguite (o ha senso eseguire) le seguenti operazioni:

| **operazione** | **Stato del processore: [utente/supervisore/entrambi]** |
|---|---|
| Invocare l'istruzione TSL (test and set lock – read-modify-write) | |
| Modificare il contenuto dello stack | |
| Abilitare le interruzioni | |
| Eseguire il codice della thread_yield() | |
| Modificare il vettore di interruzione | |
| Invocare l'istruzione SVC di invocazione di una chiamata di sistema | |

**Soluzione**

| **operazione** | **Stato del processore: [utente/supervisore/entrambi]** |
|---|---|
| Invocare l'istruzione TSL (test and set lock – read-modify-write) | Entrambi |
| Modificare il contenuto dello stack | Entrambi (nello stack del rispettivo livello) |
| Abilitare le interruzioni | Suvervisore |
| Eseguire il codice della thread_yield() | Supervisore |
| Modificare il vettore di interruzione | Supervisore |
| Invocare l'istruzione SVC di invocazione di una chiamata di sistema | Utente |

**Esercizio 2 (5 punti)**
Si consideri un processore che dispone dei seguenti registri:
- i registri speciali PC (program counter) e PS (program status) e lo stack pointer SP
- un banco di registri generali R1, R2,

Il sistema operativo realizza i thread a livello kernel, e i thread costituiscono l'unica unità di schedulazione del sistema. Come meccanismi di mutua esclusione e di sincronizzazione, il sistema offre ai thread i semafori.
Al tempo t sono presenti solo tre thread: il thread T1, in stato di esecuzione, il thread T2 in stato di pronto e il thread T3 in stato di attesa sul semaforo SEM. Nessun altro thread è presente nel sistema. Al tempo *t* il thread T1 esegue una istruzione SVC per invocare la chiamata di sistema P(SEM). Al momento dell'esecuzione dell'istruzione SVC i registri del processore, i descrittori di T1, T2 e T3 e lo stack del nucleo hanno i contenuti mostrati in tabella. Lo stack pointer del nucleo ha il valore 1016.
Il vettore di interruzione associato all'interruzione generata da SVC è 0425 e la parola di stato del nucleo è 275E.
Si chiede:

a) il contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione della prima istruzione della funzione di servizio;
b) il contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione dell'istruzione IRET con la quale termina la chiamata di sistema;
c) il contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione dell'istruzione eseguita subito dopo la IRET.

# Sistemi Operativi e Laboratorio, Prova del 4/4/2019 – versione B

| Descrittore T1 | |
|---|---|
| stato | esec. |
| PC | 2E31 |
| PS | 16F2 |
| SP | 1873 |
| R1 | 1234 |
| R2 | 16CC |
| | |

| Descrittore T2 | |
|---|---|
| stato | pronto |
| PC | B12C |
| PS | B6F2 |
| SP | BFF5 |
| R1 | B5CC |
| R2 | B000 |
| | |

| Descrittore T3 | |
|---|---|
| stato | attesa |
| PC | CC11 |
| PS | C6F2 |
| SP | CFF7 |
| R1 | C1C1 |
| R2 | C2C2 |
| | |

| Processore: registri speciali | | | | | | |
|---|---|---|---|---|---|---|
| PC | 2F00 | | PS | 16F2 | stato | utente |

| Stack del nucleo | |
|---|---|
| …… | …….. |
| 1016 | |
| 1014 | |
| 1012 | |
| 1010 | |
| 100E | |
| 100C | |
| | |
| | |

| Registri generali | |
|---|---|
| SP | AFFF |
| R1 | AA2A |
| R2 | A2CE |

**Soluzione**

a) contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione della prima istruzione della funzione di servizio:

| Descrittore T1 | |
|---|---|
| stato | |
| PC | |
| PS | |
| SP | |
| R1 | |
| R2 | |
| | |

| Descrittore T2 | |
|---|---|
| stato | |
| PC | |
| PS | |
| SP | |
| R1 | |
| R2 | |
| | |

| Descrittore T3 | |
|---|---|
| stato | |
| PC | |
| PS | |
| SP | |
| R1 | |
| R2 | |
| | |

| Processore: registri speciali | | | | | |
|---|---|---|---|---|---|
| PC | | PS | | stato | |

| Stack del nucleo | |
|---|---|
| …… | …….. |
| 1016 | |
| 1014 | |
| 1012 | |
| 1010 | |
| 100E | |
| 100C | |
| | |
| | |

| Registri generali | |
|---|---|
| SP | |
| R1 | |
| R2 | |

b) contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione dell'istruzione IRET con la quale termina la chiamata di sistema;

| Descrittore T1 | |
|---|---|
| stato | |
| PC | |
| PS | |
| SP | |
| R1 | |
| R2 | |
| | |

| Descrittore T2 | |
|---|---|
| stato | |
| PC | |
| PS | |
| SP | |
| R1 | |
| R2 | |
| | |

| Descrittore T3 | |
|---|---|
| stato | |
| PC | |
| PS | |
| SP | |
| R1 | |
| R2 | |
| | |

| Processore: registri speciali | | | | | |
|---|---|---|---|---|---|
| PC | | PS | | stato | |

| Stack del nucleo | |
|---|---|
| …… | …….. |
| 1016 | |
| 1014 | |
| 1012 | |
| 1010 | |
| 100E | |
| 100C | |
| | |
| | |

| Registri generali | |
|---|---|
| SP | |
| R1 | |
| R2 | |

c) Contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione dell'istruzione eseguita subito dopo la IRET.

| Descrittore T1 | |
|---|---|
| stato | |
| PC | |
| PS | |
| SP | |
| R1 | |
| R2 | |
| | |

| Descrittore T2 | |
|---|---|
| stato | |
| PC | |
| PS | |
| SP | |
| R1 | |
| R2 | |
| | |

| Descrittore T3 | |
|---|---|
| stato | |
| PC | |
| PS | |
| SP | |
| R1 | |
| R2 | |
| | |

| Processore: registri speciali | | | | | |
|---|---|---|---|---|---|
| PC | | PS | | stato | |

| Stack del nucleo | |
|---|---|
| …… | …….. |
| 1016 | |
| 1014 | |
| 1012 | |
| 1010 | |
| 100E | |
| 100C | |
| | |
| | |

| Registri generali | |
|---|---|
| SP | |
| R1 | |
| R2 | |

# Sistemi Operativi e Laboratorio, Prova del 4/4/2019 – versione B

**Soluzione**

a) Contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione della prima istruzione della funzione di servizio:

| Descrittore T1 | |
|---|---|
| stato | esec. |
| PC | 2E31 |
| PS | 16F2 |
| SP | 1873 |
| R1 | 1234 |
| R2 | 16CC |
| | |

| Descrittore T2 | |
|---|---|
| stato | pronto |
| PC | B12C |
| PS | B6F2 |
| SP | BFF5 |
| R1 | B5CC |
| R2 | B000 |
| | |

| Descrittore T3 | |
|---|---|
| stato | attesa |
| PC | CC11 |
| PS | C6F2 |
| SP | CFF7 |
| R1 | C1C1 |
| R2 | C2C2 |
| | |

| Processore: registri speciali | | | | | |
|---|---|---|---|---|---|
| PC | 0425 | PS | 275E | stato | superviso-re |

| Stack del nucleo | |
|---|---|
| …… | …….. |
| 1016 | 2F00 |
| 1014 | 16F2 |
| 1012 | AFFF |
| 1010 | |
| 100E | |
| 100C | |
| | |
| | |

| Registri generali | |
|---|---|
| SP | 1010 |
| R1 | AA2A |
| R2 | A2CE |

b) Contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione dell'istruzione IRET con la quale termina la chiamata di sistema;

| Descrittore T1 | |
|---|---|
| stato | attesa |
| PC | 2F00 |
| PS | 16F2 |
| SP | AFFF |
| R1 | AA2A |
| R2 | A2CE |
| | |

| Descrittore T2 | |
|---|---|
| stato | Esecuz. |
| PC | B12C |
| PS | B6F2 |
| SP | BFF5 |
| R1 | B5CC |
| R2 | B000 |
| | |

| Descrittore T3 | |
|---|---|
| stato | attesa |
| PC | CC11 |
| PS | C6F2 |
| SP | CFF7 |
| R1 | C1C1 |
| R2 | C2C2 |
| | |

| Processore: registri speciali | | | | | |
|---|---|---|---|---|---|
| PC | 0425+? | PS | 275E | stato | superviso-re |

| Stack del nucleo | |
|---|---|
| …… | …….. |
| 1016 | B12C |
| 1014 | B6F2 |
| 1012 | BFF5 |
| 1010 | |
| 100E | |
| 100C | |
| | |
| | |

| Registri generali | |
|---|---|
| SP | 1010 |
| R1 | B5CC |
| R2 | B000 |

c) Contenuto dei descrittori, dei registri generali e speciali, dello stack del nucleo e lo stato del processore durante la fase di estrazione dell'istruzione eseguita subito dopo la IRET.

| Descrittore T1 | |
|---|---|
| stato | attesa |
| PC | 2F00 |
| PS | 16F2 |
| SP | AFFF |
| R1 | AA2A |
| R2 | A2CE |
| | |

| Descrittore T2 | |
|---|---|
| stato | Esecuz. |
| PC | B12C |
| PS | B6F2 |
| SP | BFF5 |
| R1 | B5CC |
| R2 | B000 |
| | |

| Descrittore T3 | |
|---|---|
| stato | attesa |
| PC | CC11 |
| PS | C6F2 |
| SP | CFF7 |
| R1 | C1C1 |
| R2 | C2C2 |
| | |

| Processore: registri speciali | | | | | |
|---|---|---|---|---|---|
| PC | B12C | PS | B6F2 | stato | utente |

| Stack del nucleo | |
|---|---|
| …… | …….. |
| 1016 | |
| 1014 | |
| 1012 | |
| 1010 | |
| 100E | |
| 100C | |
| | |
| | |

| Registri generali | |
|---|---|
| SP | BFF5 |
| R1 | B5CC |
| R2 | B000 |

**Esercizio 3 (5 punti)**

Relativamente alle variabili di condizione, spiegare (in max. 5 righe) cosa sono e quali sono le differenze tra le semantiche MESA e HOARE.

_____

_____

_____

_____

_____

Un aeroporto dispone di `n` gate e di un'unica pista di atterraggio. Quando un aeromobile intende atterrare inizia una procedura di richiesta di atterraggio. In questa procedura l'aeromobile va preliminarmente in stato di *attesa gate* nel quale richiede la disponibilità di un gate alla torre di controllo. L'aeromobile resta in questo stato finché la torre di controllo non individua un gate libero e glielo assegna (i gate vengono liberati dagli eventuali aeromobili in partenza, il cui codice non è richiesto in questo esercizio). A questo punto l'aeromobile passa in stato *attesa pista* fintanto che la pista non si libera per l'atterraggio, ed infine atterra. Completato l'atterraggio l'aeromobile di dirige al gate e libera la pista.

L'intera procedura avviene quindi secondo la seguente sequenza:

```
richiestaAtterraggio(idAeromobile)
<atterraggio>
liberaPista()
```

Ipotizzando una gestione interamente automatica dell'aeroporto e degli aeromobili, ogni aeromobile è associato ad un thread che ne gestisce l'intera procedura di atterraggio (denotiamo tali thread $A_1,\ldots,A_m$).
Si chiede di completare le procedure `richiestaAtterraggio(idAeromobile)` e `liberaPista()` assumendo che i thread $A_1,\ldots,A_m$ utilizzino le seguenti variabili condivise per coordinare le varie fasi dell'atterraggio:

- `mutex` – per gestire la mutua esclusione tra aeromobili;
- `attesaGate` – variabile di condizione sulla quale si sospendono gli aeromobili che non hanno ancora ottenuto un gate
- `attesaPista` – variabile di condizione sulla quale si sospendono gli aeromobili in attesa della disponibilità della pista per l'atterraggio;

Lo stato del sistema è inoltre rappresentato dalle seguenti variabili:

- `waitGate` (inizializzato a `0`): è il numero di aerei in attesa di un gate libero;
- `freeGate` : è il numero di gate liberi;
- `waitLand` (inizializzata a `0`): è il numero di aerei in attesa della pista per l'atterraggio;
- `freeLane` (inizializzata a `True`): è `True` se la pista è libera, è `False` altrimenti.

Si chiede di completare la seguente soluzione assumendo che la semantica delle variabili di condizione sia di tipo MESA.

```
richiestaAtterraggio(Ai) {

        _____________________________
        waitGate ++;
        while (freeGate == 0)

                _____________________________
        waitGate --;
        freeGate--;
        waitLand++;
        while (!freeLane)

                _____________________________
        waitLand--;
        freeLane=FALSE;

        _____________________________
}
liberaPista() {
        _____________________________

        freeLane =TRUE;

        if (waitLand >0) _____________________________

        _____________________________
}
```

**Soluzione**

```
richiestaAtterraggio(Ai) {
        mutex.Acquire();
        waitGate ++;
        while (freeGate == 0)
                attesaGate.wait(mutex);
        waitGate --;
        freeGate--;
        waitLand++;
        while (!freeLane)
                attesaPista.wait(mutex);
        waitLand--;
        freeLane=FALSE;

        mutex.Release();
}
liberaPista() {
```

```
        mutex.Acquire();
        freeLane =TRUE;
        if (waitLand >0) attesaPista.signal();
        mutex.Release();
}
```

**Esercizio 4 (5 punti)**

Dire a quale livello (nucleo e/o utente) i semafori possono essere implementati, motivando la risposta. (max. 5 righe):

_____

_____

_____

_____

_____

Si considerino due thread consumatori P1 e P2 e due thread produttori C1 e C2 che cooperano utilizzando un buffer condiviso a cinque posizioni, utilizzando il seguente codice:

| Codice consumatori | Codice produttori |
|---|---|
| while TRUE {<br>1. P(empty);<br>2. P(mutex);<br><preleva un dato dal buffer><br>3. V(mutex);<br>4. V(full);<br><*usa dato*><br>} | while TRUE {<br><*prepara nuovo dato*><br>a. P(full);<br>b. P(mutex);<br><deposita dato nel buffer><br>c. V(mutex);<br>d. V(empty);<br>} |

Ipotizzando che il thread in esecuzione resti in tale stato finchè non si interrompe spontaneamente (il sistema operativo non forza il prerilascio del processore), si considerino i seguenti casi (da considerare **in alternativa**):

**caso 1**: Ad un certo istante di tempo lo stato del sistema è il seguente:

- P1 è **in esecuzione** e deve eseguire la linea 1;
- P2 è pronto per eseguire la linea 1;
- C1 è pronto per eseguire la linea a;
- C2 è pronto per eseguire la linea d;

Mostrare lo stato del sistema al termine dell'esecuzione di ogni linea di codice numerata eseguita da P1, per 5 linee di codice eseguite o fino a che P1 si sospende. In ogni riga mostrare lo stato risultante dopo l'esecuzione dell'istruzione corrispondente.

| Stato P1 | Linea eseguita | Empty (valore, coda) | Full (valore, coda) | Mutex (valore, coda) |
|---|---|---|---|---|
| **Esecuzione** | **Prima di eseguire 1** | **1, –** | **3, –** | **1, –** |
| | **1** | | | |
| | **2** | | | |
| | **3** | | | |
| | **4** | | | |
| | **1** | | | |

**caso 2**: Ad un certo istante di tempo lo stato del sistema è il seguente:

- P1 è pronto per eseguire la linea 1;
- P2 è pronto per eseguire la linea 4;
- C1 è **in esecuzione** e deve eseguire la linea a;
- C2 è pronto per eseguire la linea a;

Dire cosa come cambia lo stato del sistema a partire dal tempo t, per 5 linee di codice eseguite o fino all'istante in cui il processo C1 si sospende. In ogni riga mostrare lo stato risultante dopo l'esecuzione dell'istruzione corrispondente.

| Stato C1 | Linea eseguita | Empty (valore, coda) | Full (valore, coda) | Mutex (valore, coda) |
|---|---|---|---|---|
| **Esecuzione** | **prima di eseguire a** | **1, –** | **3, –** | **1, –** |

| | | | | |
|---|---|---|---|---|
| | **a** | | | |
| | **b** | | | |
| | **c** | | | |
| | **d** | | | |
| | **a** | | | |

**Soluzione**

**caso 1**:

| Stato P1 | Linea eseguita | Empty (valore, coda) | Full (valore, coda) | Mutex (valore, coda) |
|---|---|---|---|---|
| **Esecuzione** | **Prima di eseguire 1** | **1, –** | **3, –** | **1, –** |
| Esecuzione | **1** | 0, – | 3, – | 1, – |
| Esecuzione | **2** | 0, – | 3, – | 0, – |
| Esecuzione | **3** | 0, – | 3, – | 1, – |
| Esecuzione | **4** | 0, – | 4, – | 1, – |
| Attesa | **1** | 0, P1 | 4, – | 1, – |

**caso 2**:

| Stato C1 | Linea eseguita | Empty (valore, coda) | Full (valore, coda) | Mutex (valore, coda) |
|---|---|---|---|---|
| **Esecuzione** | **prima di eseguire a** | **1, –** | **3, –** | **1, –** |
| Esecuzione | **a** | 1, – | 2, – | 1, – |
| Esecuzione | **b** | 1, – | 2, – | 0, – |
| Esecuzione | **c** | 1, – | 2, – | 1, – |
| Esecuzione | **d** | 2, – | 2, – | 1, – |
| Esecuzione | **a** | 2, – | 1, – | 1, – |

**Esercizio 5 (3 punti)**

In un sistema con risorse R1, R2, R3, R4, R5, R6 (tutte di molteplicità 3), sono presenti i processi P2, P1 e P3 che inizialmente non possiedono risorse e successivamente avanzano senza interagire reciprocamente e alternandosi nello stato di esecuzione con velocità arbitrarie.

Nel corso della propria esistenza, ciascun processo esegue una propria sequenza di richieste, che possono intercalarsi in modo arbitrario con quelle degli altri processi. Dopo aver ottenuto e utilizzato le risorse che richiede, ogni processo termina rilasciando tutte le risorse ottenute.

Si consideri la sequenza di richieste sotto riportate:

| Processo | Prima richiesta | Seconda richiesta | Terza richiesta | Quarta richiesta | Terminazione |
|---|---|---|---|---|---|
| P1 | 1 istanza di R4 | 2 istanza di R6 | 2 istanza di R2 | 1 istanza di R5 | Rilascia tutto |
| P2 | 1 istanza di R6 | 1 istanza di R2 | 2 istanza di R5 | 1 istanza di R1 | Rilascia tutto |
| P3 | 2 istanza di R3 | 2 istanza di R2 | 1 istanza di R1 | - | Rilascia tutto |

Dire se i processi evitano la possibilità di stallo e motivare la risposta.

**Soluzione**

I processi evitano lo stallo [SI/NO]? _____

Motivazione [se evitano lo stallo spiegare perché. Se non evitano lo stallo mostrare una sequenza che li porta in stallo]:

_____

_____

_____

_____

**Soluzione**

I processi evitano lo stallo perché la sequenza richieste di allocazione di risorse è ordinata. Un possibile ordine è:

R4, R3, R6, R2, R5, R1

## Esercizio 6 (5 punti)

In un sistema che utilizza l'algoritmo del banchiere per prevenire lo stallo, spiegare a quale livello [kernel/utente] questo algoritmo deve essere implementato e in quali occasioni viene eseguito (max. 5 righe):

_____

_____

_____

_____

_____

Un sistema con 5 processi (A, B, C, D, E) e risorse dei tipi R1, R2, R3, R4, rispettivamente di molteplicità [6, 8, 5, 7], utilizza l'algoritmo del banchiere per evitare lo stallo. Il sistema ha raggiunto lo stato (sicuro) mostrato nelle tabelle seguenti.

| Assegnazione attuale | | | | |
|---|---|---|---|---|
| | R1 | R2 | R3 | R4 |
| A | | 2 | 2 | 3 |
| B | | | 1 | 1 |
| C | | | | 2 |
| D | 2 | 2 | | |
| E | 3 | 2 | | |

| Esigenza residua | | | | |
|---|---|---|---|---|
| | R1 | R2 | R3 | R4 |
| A | 2 | 2 | 1 | 0 |
| B | 5 | 6 | 4 | 1 |
| C | 3 | 3 | 2 | 3 |
| D | 1 | 2 | 2 | 0 |
| E | 1 | 5 | 3 | 6 |

| Molteplicità | | | |
|---|---|---|---|
| R1 | R2 | R3 | R4 |
| 6 | 8 | 5 | 7 |

| Disponibilità | | | |
|---|---|---|---|
| 1 | 2 | 2 | 1 |

Si considerino ora i seguenti casi (**in alternativa**):

a) Il processo D richiede due istanze della risorsa R3
b) Il processo B richiede un'istanza della risorsa R4

In ognuno dei due casi, dire se la risorsa viene assegnata dal sistema operativo e quale è lo stato del processo interessato dopo che l'algoritmo del banchiere ha analizzato la richiesta di risorse.

### Soluzione

Stato raggiunto dopo l'ipotetica assegnazione di due istanze di R3 al processo D:

| Assegnazione attuale | | | | |
|---|---|---|---|---|
| | R1 | R2 | R3 | R4 |
| A | | | | |
| B | | | | |
| C | | | | |
| D | | | | |
| E | | | | |

| Esigenza residua (dopo l'assegnazione attuale) | | | | |
|---|---|---|---|---|
| | R1 | R2 | R3 | R4 |
| A | | | | |
| B | | | | |
| C | | | | |
| D | | | | |
| E | | | | |

| Molteplicità | | | |
|---|---|---|---|
| R1 | R2 | R3 | R4 |
| 6 | 8 | 5 | 7 |

| Disponibilità | | | |
|---|---|---|---|
| | | | |

a1) Il processo _____ può/non può* terminare; la disponibilità di {R1, R2, R3, R4} diviene _____
a2) Il processo _____ può/non può* terminare; la disponibilità di {R1, R2, R3, R4} diviene _____
a3) Il processo _____ può/non può* terminare; la disponibilità di {R1, R2, R3, R4} diviene _____
a4) Il processo _____ può/non può* terminare; la disponibilità di {R1, R2, R3, R4} diviene _____
a5) Il processo _____ può/non può* terminare; la disponibilità di {R1, R2, R3, R4} diviene _____

* Cancellare ciò che non si applica

Stato sicuro? _____

Di conseguenza: risorsa assegnata? _____ Stato di D? _____

Stato raggiunto dopo l'ipotetica assegnazione di un'istanza di R4 al processo B:

| Assegnazione attuale | | | | |
|---|---|---|---|---|
| | R1 | R2 | R3 | R4 |

| Esigenza residua (dopo l'assegnazione attuale) | | | | |
|---|---|---|---|---|
| | R1 | R2 | R3 | R4 |

| Molteplicità | | | |
|---|---|---|---|
| R1 | R2 | R3 | R4 |

## Sistemi Operativi e Laboratorio, Prova del 4/4/2019 – versione B

| | | | | |
|---|---|---|---|---|
| A | | | | |
| B | | | | |
| C | | | | |
| D | | | | |
| E | | | | |

| | | | | |
|---|---|---|---|---|
| A | | | | |
| B | | | | |
| C | | | | |
| D | | | | |
| E | | | | |

| | | | |
|---|---|---|---|
| 6 | 8 | 5 | 7 |

| Disponibilità | | | |
|---|---|---|---|
| | | | |

b1) Il processo _____ può/non può* terminare; la disponibilità di {R1, R2, R3, R4} diviene _____
b2) Il processo _____ può/non può* terminare; la disponibilità di {R1, R2, R3, R4} diviene _____
b3) Il processo _____ può/non può* terminare; la disponibilità di {R1, R2, R3, R4} diviene _____
b4) Il processo _____ può/non può* terminare; la disponibilità di {R1, R2, R3, R4} diviene _____
b5) Il processo _____ può/non può* terminare; la disponibilità di {R1, R2, R3, R4} diviene _____
* Cancellare ciò che non si applica
Stato sicuro? _____
Di conseguenza: risorsa assegnata? _____ Stato di B? _____

**Soluzione**
Stato raggiunto dopo l'ipotetica assegnazione di due istanze di R3 al processo D:

| Assegnazione attuale | | | | |
|---|---|---|---|---|
| | R1 | R2 | R3 | R4 |
| A | | 2 | 2 | 3 |
| B | | | 1 | 1 |
| C | | | | 2 |
| D | 2 | 2 | 2 | |
| E | 3 | 2 | | |

| Esigenza residua | | | | |
|---|---|---|---|---|
| | R1 | R2 | R3 | R4 |
| A | 2 | 2 | 1 | 0 |
| B | 5 | 6 | 4 | 1 |
| C | 3 | 3 | 2 | 3 |
| D | 1 | 2 | 0 | 0 |
| E | 1 | 5 | 3 | 6 |

| Molteplicità | | | |
|---|---|---|---|
| R1 | R2 | R3 | R4 |
| 6 | 8 | 5 | 7 |

| Disponibilità | | | |
|---|---|---|---|
| 1 | 2 | 0 | 1 |

a1) Il processo D può terminare; la disponibilità di {R1, R2, R3, R4} diviene {3,4,2,1}
a2) Il processo A può terminare ; la disponibilità di {R1, R2, R3, R4} diviene {3,6,4,4}
a3) Il processo C può terminare; la disponibilità di {R1, R2, R3, R4} diviene {3,6,4,6}
a4) Il processo E può terminare; la disponibilità di {R1, R2, R3, R4} diviene {6,8,4,6}
a5) Il processo B può terminare; la disponibilità di {R1, R2, R3, R4} diviene {6,8,5,7}

Stato sicuro? **SI**
Risorsa assegnata? **SI**
Stato di D? **ESECUZIONE**

Stato raggiunto dopo l'ipotetica assegnazione di un'istanza di R4 al processo B:

| Assegnazione attuale | | | | |
|---|---|---|---|---|
| | R1 | R2 | R3 | R4 |
| A | | 2 | 2 | 3 |
| B | | | 1 | 2 |
| C | | | | 2 |
| D | 2 | 2 | | |
| E | 3 | 2 | | |

| Esigenza residua (dopo l'assegnazione attuale) | | | | |
|---|---|---|---|---|
| | R1 | R2 | R3 | R4 |
| A | 2 | 2 | 1 | 0 |
| B | 5 | 6 | 4 | 0 |
| C | 3 | 3 | 2 | 3 |
| D | 1 | 2 | 2 | 0 |
| E | 1 | 5 | 3 | 6 |

| Molteplicità | | | |
|---|---|---|---|
| R1 | R2 | R3 | R4 |
| 6 | 8 | 5 | 7 |

| Disponibilità | | | |
|---|---|---|---|
| 1 | 2 | 2 | 0 |

a1) Il processo D può terminare; la disponibilità di {R1, R2, R3, R4} diviene {3,4,2,0}
a2) Il processo A può terminare ; la disponibilità di {R1, R2, R3, R4} diviene {3,6,4,3}
a3) Il processo C può terminare ; la disponibilità di {R1, R2, R3, R4} diviene {3,6,4,5}
i processi B e E non possono terminare

Stato sicuro? **NO**
Risorsa assegnata? **NO**
Stato di di B? **ATTESA**

**Esercizio 7 (3 punti)**
Si consideri il seguente frammento di codice:

```
…
a = fork();
if (a <= 0) {
    printf(“D”);
    exit(1);
} else {
    b=fork();
    if (b > 0) {
            c= wait(&status);
            printf(“E”);
    }
    else printf(“B”); }
}
printf(“F”);
…
```

Il processo che esegue l’intero frammento è quello che esegue le due fork, cioè il padre.
Il processo generato con la prima fork è il primo figlio.
Il processo generato con la seconda fork è il secondo figlio.
Dire cosa stampano il padre e i due figli considerando i seguenti casi di successo e fallimento delle fork:

a) Se entrambe le fork hanno successo:
Il padre stampa: _____
Il primo figlio stampa: _____
Il secondo figlio stampa: _____

b) Se la prima fork ha successo e la seconda fallisce:
Il padre stampa: _____
Il primo figlio stampa: _____
Il secondo figlio stampa: _____

c) Se la prima fork fallisce:
Il padre stampa: _____
Il primo figlio stampa: _____
Il secondo figlio stampa: _____

**Soluzione**
a) Se entrambe le fork hanno successo:
Il padre stampa: EF
Il primo figlio stampa: D
Il secondo figlio stampa: BF

b) Se la prima fork ha successo e la seconda fallisce:
Il padre stampa: BF
Il primo figlio stampa: D
Il secondo figlio non esiste.

c) Se la prima fork fallisce:

Il padre stampa: D
Il primo figlio non esiste.
Il secondo figlio non esiste.

**Sistemi Operativi e Laboratorio, Prova del 4/4/2019 – versione B**

Nome: _____ Cognome: _____ Matricola:_____ corso:____

**Esercizio 8 (4 punti)**:
Dati i seguenti 3 files (myutil.h, myutil.c e myprog.c):

myutil.h

```
#if !defined(MUTIL_H_)
#define MUTIL_H_
#include<stdlib.h>
#include<stdio.h>
int G(int);
#endif
```

myutil.c

```
#include <myutil.h>
int S=15;
int G(int x) { return S -=x; }
```

myprog.c

```
#include <myutil.h>
int main() {
  extern int S;
  int s1=G(3);
  int s2=G(4);
  S +=s1; S -=s2;
  printf("S=%d\n", S);
  return 0;
}
```

completare il seguente mini **Makefile** per generare l'eseguibile **myprog**:

```
CC=gcc
AR=ar
CFLAGS= -std=c99 -Wall -g
ARFLAGS=rvs
INCLUDE_DIR=./includes
LIB_DIR=./libs
TARGET=myprog
.PHONY= ……                                  # indicare i phony targets

%.o: %.c
        $(CC)  ……. -c -o $@ $<              # completare la regola

…….: ……                                    # completare la regola
       $(CC) …… -o $@ $<  ……

…….: myutil.o ……...                         # completare la regola
      $(AR) $(ARFLAGS) $@ $<

cleanall:
    \rm -f *.o *~ $(LIB_DIR)/libmyprog.a $(TARGET)
```

Dire se la funzione G è rientrante o meno motivando (brevemente) la risposta.

_____

Qual è l'output fornito dal programma ./myprog ?

_____

**SOLUZIONE:**

```
CC=gcc
AR=ar
CFLAGS= -std=c99 -Wall -g
ARFLAGS=rvs
INCLUDE_DIR=./includes
LIB_DIR=./libs
.PHONY= cleanall

%.o: %.c
	$(CC) $(CFLAGS) -I$(INCLUDE_DIR) -c -o $@ $<

myprog: myprog.o $(LIB_DIR)/libmyprog.a
	$(CC) $(CFLAGS) -I$(INCLUDE_DIR) -o $@ $< -L$(LIB_DIR) -lmyprog

$(LIB_DIR)/libmyprog.a: myutil.o $(INCLUDE_DIR)/myutil.h
	$(AR) $(ARFLAGS) $@ $<

cleanall:
	\rm -f *.o *~ myprog $(LIB_DIR)/libmyprog.a
```

La funzione G non è rientrante perché legge e scrive la variabile globale S.

L'output di ./myprog è: 12

**Esercizio 9 (2 punti)**:

a) Dire che cosa è una libreria e quali sono le principali differenze tra una libreria statica (.a) ed una dinamica (.so) (massimo 5 righe).

_____

_____

_____

_____

_____

_____

b) Riempire la seguente tabella descrivendo sinteticamente il comando oppure il file riportato nella parte sinistra della tabella:

| File o comando | Descrizione |
|---|---|
| */bin/*ls <dir> | Esegue il listing della directory <dir> |
| */usr/bin/*find | |
| */bin*/bash | |
| */bin*/chmod | |
| */bin/*mkdir | |

| | |
|---|---|
| */etc*/passwd | |

**SOLUZIONE:**

b)

| **Directory** | **Descrizione** |
|---|---|
| */bin/*ls <dir> | Esegue il listing della directory <dir> |
| */usr/bin*/find | Comando di ricerca file in modo ricorsivo a partire da una directory |
| */bin*/bash | Interprete della shell Bash |
| */bin*/chmod | Comando che permette di cambiare i permessi di files e directories |
| */bin/*mkdir | Permette di creare una directory |
| */etc*/passwd | E' il file contenente la lista degli utenti del sistema. |