Quesito 1 (punti 5). Si consideri un elaboratore dotato di un processore con frequenza di *clock* fissata a 1 MHz. Si assuma che un ciclo di *clock* sia sufficiente al processore per eseguire una istruzione quando essa non includa riferimenti a pagine di memoria diverse da quella correntemente acceduta. Si assuma inoltre che l'elaboratore considerato sia dotato di un sistema di paginazione per la gestione della memoria virtuale e che valgano le seguenti ipotesi:

- l'accesso a una pagina di memoria diversa dalla corrente comporta un ulteriore costo temporale di 1 microsecondo;
- ciascuna pagina di memoria è ampia 1.000 *Byte*;
- il disco fisso rotea a una velocità di 6.000 RPM (giri al minuto);
- il trasferimento tra disco e memoria avviene a 1.000.000 *Byte* al secondo;
- il tempo medio per spostare la testina del disco dalla posizione corrente fin sopra la traccia dove si trova il punto di scrittura/lettura è di 10 ms;
- un blocco su disco corrisponde in dimensione esattamente a una pagina di memoria virtuale;
- 99% delle istruzioni eseguite fanno riferimento alla pagina corrente;
- 80% delle pagine accedute (diverse dalla corrente) si trova già in memoria;
- quando una nuova pagina deve essere caricata in memoria, nel 50% dei casi quella rimpiazzata era stata modificata.

Si calcoli il tempo effettivo medio di esecuzione di ciascuna istruzione su tale elaboratore assumendo che il sistema stia eseguendo un unico processo e che il processore rimanga inattivo (stato *idle*) durante i trasferimenti di dati.

Quesito 2 (punti 7). Cinque processi *batch*, identificati dalle lettere $A - E$ rispettivamente, arrivano all'elaboratore agli istanti $0, 2, 3, 6, 10$ rispettivamente. Tali processi hanno un tempo di esecuzione stimato di $3, 6, 4, 5, 1$ unità di tempo rispettivamente. Per ognuna delle seguenti politiche di ordinamento:

1. RR (divisione di tempo, senza priorità e con quanto di tempo di ampiezza 2)
2. RR (divisione di tempo, con priorità e prerilascio, e quanto di tempo di ampiezza 2)
3. $\mathsf{SJF}_{\mathtt{SRTN}}$ (senza considerazione di valori di priorità espliciti[1] e con prerilascio)

determinare, trascurando i ritardi dovuti allo scambio di contesto: (i) il tempo medio di risposta; (ii) il tempo medio di attesa; (iii) il tempo medio di *turn around*.
Ove la politica di ordinamento in esame consideri i valori di priorità, tali valori, mantenuti staticamente per l'intera durata dell'esecuzione, sono rispettivamente: $2, 3, 5, 3, 2$ (con 5 valore maggiore).
Nel caso di arrivi simultanei di processi allo stato di pronto, fatta salva l'eventuale considerazione del rispettivo valore di priorità, si dia la precedenza ai processi usciti dallo stato di esecuzione rispetto a quelli appena arrivati.

Quesito 3 (punti 4). Si consideri un sistema dotato di memoria virtuale, con memoria fisica divisa in 8 *page frame*, condivisa da 4 processi $A - D$ contemporaneamente attivi. Si supponga che all'istante 100 lo stato della memoria sia il seguente (con il *bit* di riferimento avente significato: 1 = pagina riferita; 0 non riferita):

processo	pag. logica	pag. fisica	istante di caricamento	*bit* di riferimento
A	0	7	50	1
A	1	6	37	0
B	5	5	30	1
B	3	4	97	0
C	2	0	15	1
C	5	2	70	0
C	9	1	92	0
D	8	3	27	0

Tabella 1: Situazione all'istante 100.

Si supponga che il sistema utilizzi un algoritmo di rimpiazzo di tipo *second chance*(globale). All'istante 100, come riportato in tabella 1, e rappresentato in forma compatta, la lista delle pagine accedute è:

C2(15,1) - D8(27,0) - B5(30,1) - A1(37,0) - A0(50,1) - C5(70,0) - C9(92,0) - B3(97,0)

[1] Esclusi ovviamente i valori di priorità impliciti determinati dalla durata (residua) dei processi.

dove C2(15,1) sia il primo elemento della lista; in tale elemento, C2 indica che si tratta della pagina logica 2 del processo C, mentre (15,1) indica che tale pagina è stata caricata in memoria all'istante 15 e che il suo *bit* di riferimento vale 1.

Si consideri la seguente sequenza di esecuzione:

1. all'istante 101, il processo C riferisce la pagina logica 5
2. all'istante 105, il processo A riferisce la pagina logica 9

Assumendo che, a parte quelle sopra elencate, non intervengano altre attività capaci di modificare i *bit* di riferimento, si scriva la lista delle pagine accedute aggiornata dopo aver eseguito, in successione, la sequenza suddetta nell'ordine dato.

Quesito 4 (punti 4). Si consideri una variante *FAT-15* dell'architettura di *file system* nota come *FAT-16*, nella quale l'unica differenza dalla versione base sia che 1 *bit* dell'indice FAT non sia utilizzabile, tutti gli altri attributi di architettura rimanendo inalterati. Si specifichi la dimensione massima di *file* possibile con tale variante, e la dimensione massima di partizione.

Quesito 5 (punti 6). Discutere concisamente le differenze che intercorrono tra *deadlock* e *starvation*. Si illustri poi la definizione di *race condition*. Infine si proponga un esempio concreto di ciascuna delle tre situazioni.

Quesito 6 (punti 6). Illustrare i passi della sequenza di *boot* (inizializzazione) di un sistema operativo della famiglia UNIX/Linux. Discutere concisamente la necessità di ciascun passo in tale sequenza; in altre parole, valutare se e quali di tali passi possano essere omessi e, se del caso, sotto quali ipotesi.

Soluzione 1 (punti 5). Il processore a 1 MHz considerato dal quesito riesce a eseguire una istruzione ogni microsecondo qualora non vi siano riferimenti a pagine di memoria diverse da quella correntemente acceduta. L'esecuzione di ogni istruzione richiede quindi almeno 1 μs.
Per ipotesi, l'1% delle istruzioni richiede 1 μs in più per accedere a un'area di memoria diversa da quella corrente; di queste, il 20% richiede tempo ulteriore per leggere dal disco un pagina di memoria virtuale e, nel 50% dei casi, la pagina rimpiazzata dovrà essere copiata su disco, aumentando quindi il tempo di esecuzione dell'istruzione corrispondente.
Il tempo medio di esecuzione comprenderà dunque:

- il tempo di esecuzione su processore: 1 μs per tutte le istruzioni;
- il tempo di accesso ad area di memoria diversa dalla corrente: 1 μs per l'1% delle istruzioni;
- il tempo di lettura da disco della pagina riferita, comprensivo dei tempi medi di posizionamento, rivoluzione e trasmissione, per il 20% delle istruzioni che fanno riferimento a pagine diverse dalla corrente;
- il tempo di scrittura su disco della pagina rimpiazzata: come per il punto precedente, ma solo nel 50% dei casi in cui si deve accedere al disco;

Il disco rotea a 6.000 giri al minuto, corrispondenti a 100 giri/secondo; una rivoluzione completa avviene quindi in 10 ms. Statisticamente, quando la testina del disco si muove per leggere o scrivere su un'altra traccia, si troverà in media a mezzo giro di disco di distanza dalla posizione cercata, aggiungendo quindi il corrispondente ritardo di 5 ms in media a ogni spostamento di testina. A ogni lettura o scrittura su disco occorre dunque aggiungere il tempo di movimento della testina (per ipotesi, in media 10 ms), il tempo di rotazione del disco per riportare la testina al punto iniziale dove leggere o scrivere (come detto, in media 5 ms) e infine il tempo di trasferimento della pagina. Quest'ultimo valore è pari a: 1.000 B/1.000.000 B/s = 1 ms.
In totale quindi, ogni accesso a disco comporta un ulteriore ritardo di: 5 ms + 10 ms + 1 ms = 16 ms.
In conclusione, il tempo medio di esecuzione di un'istruzione risulta dunque essere pari a:

$$1\mu s + 0,01 \times [1\mu s + 0,2 \times (16ms + 0,5 \times 16ms)] = 49,01\mu s.$$

Soluzione 2 (punti 7).

- RR (divisione di tempo, senza priorità e con quanto di tempo di ampiezza 2)

```
            0123456789012345678
processo A  AAA                     LEGENDA DEI SIMBOLI
processo B  --bBBbbBBbbbbBB         - non ancora arrivato
processo C  ---ccCCccccCC           x (minuscolo) attesa
processo D  ------dddDDdddddDDD     X (maiuscolo) esecuzione
processo E  ----------eeeeeE        . coda vuota

CPU         AAABBCCBBDDCCBBEDDD
coda        ..bccbbddccbbeed...
            ......dccbbeedd....
            ..........edd......
```

processo	tempo di risposta	attesa	*turn-around*
A	0	0	0+3= 3
B	1	7	7+6=13
C	2	6	6+4=10
D	3	8	8+5=13
E	5	5	5+1= 6
medie	2,20	5,20	9,00

- RR (divisione di tempo, con priorità e prerilascio, e quanto di tempo di ampiezza 2)

```
            0123456789012345678
processo A  AAaaaaaaaaaaaaaaaA      LEGENDA DEI SIMBOLI
processo B  --BbbbbBBbbBBbbB        - non ancora arrivato
```

```
processo C  ---CCCC                  x (minuscolo) attesa
processo D  ------dddDDddDDdD        X (maiuscolo) esecuzione
processo E  ----------eeeeeeeeE      . coda vuota

CPU         AABCCCCBBDDBBDDBDAE
coda        ..abbbbddbbddbbdae.
            ...aaadaaaaaaaae..
            ......a...eeeeee...
```

processo	tempo di risposta	tempo di attesa	tempo di *turn-around*
A	0	15	15+3=18
B	0	8	8+6=14
C	0	0	0+4= 4
D	3	6	6+5=11
E	8	8	8+1= 9
medie	2,20	7,40	11,20

- $\mathrm{SJF}_{\mathrm{SRTN}}$ (senza considerazione di valori di priorità espliciti e con prerilascio)

```
            0123456789012345678
processo A  AAA                      LEGENDA DEI SIMBOLI
processo B  --bbbbbbbbbbbBBBBBB     - non ancora arrivato
processo C  ---CCCC                  x (minuscolo) attesa
processo D  ------dDDDdDD            X (maiuscolo) esecuzione
processo E  ----------E              . coda vuota

CPU         AAACCCCDDDEDDBBBBBB
coda        ..bbbbdbbbdbb......
            ......b...b........
```

processo	tempo di risposta	tempo di attesa	tempo di *turn-around*
A	0	0	0+3= 3
B	11	11	11+6=17
C	0	0	0+4= 4
D	1	2	2+5= 7
E	0	0	0+1= 1
medie	2,40	2,60	6,40

Soluzione 3 (punti 4).
Esecuzione attività 1 (istante 101): La pagina C5 è già in memoria (nella pagina fisica 2) e viene quindi marcata come riferita. La lista risultante è dunque:

C2(15,1) - D8(27,0) - B5(30,1) - A1(37,0) - A0(50,1) - C5(70,1) - C9(92,0) - B3(97,0)

Esecuzione attività 2 (istante 105): La pagina logica A9 non si trova in memoria, l'algoritmo di rimpiazzo *second chance* (globale) esamina C2: siccome è stata riferita, essa viene inserita in coda alla lista e il *bit* di riferimento viene posto a 0; dopodiché, si esamina D8, che non risulta riferita e viene dunque eliminata. La lista risultante è dunque:

B5(30,1) - A1(37,0) - A0(50,1) - C5(70,1) - C9(92,0) - B3(97,0) - C2(15,0) - A9(105,1)

Soluzione 4 (punti 4). Come sappiamo dalla diapositiva 243 del corso dell'a.a. corrente, l'indice X dell'architettura di *file system FAT-X* denota il numero di *bit* utilizzati per esprimere l'indice di blocco. Per le ipotesi del quesito abbiamo $X = 15$, il che significa che l'intera partizione potrà constare di non più di $2^{15} = 32.768$ blocchi. Poiché il quesito richiede che tutte le altre caratteristiche dell'architettura in esame restino uguali allo standard *FAT-16*, i blocchi su disco avranno l'ampiezza massima prevista in *FAT-16*, ovvero 32 KB $= 2^5 \times 2^{10}$ B $= 2^{15}$ B. La dimensione massima di *file*, così come la dimensione massima di partizione, saranno dunque pari a: 2^{15} blocchi $\times\ 2^{15}$ B/blocco $= 2^{30}$ B $= 1$ GB, ovvero metà della dimensione massima ottenibile con *FAT-16*, che come sappiamo era pari a: 2 GB.

Soluzione 5 (punti 6). Come più volte visto a lezione, la situazione di *race condition* è uno dei tre principali nemici della concorrenza. Tale situazione (il cui nome gergale sta a significare "stato di [incertezza da] competizione") si verifica quando, laddove più *thread* competano per l'accesso R/W a una stessa risorsa logica, lo stato finale della risorsa dipenda dalle scelte di prerilascio effettuate dal sistema operativo e non dalla logica applicativa dei *thread*. Il problema viene curato regolando l'accesso alla risorsa condivisa mediante strutture come semafori o *monitor*. Un esempio di situazione esposta al rischio di *race condition* è illustrato dalla diapositiva 24 del corso dell'a.a. corrente.
Deadlock e *starvation* sono le altre due principali situazioni di pericolo in ambito di concorrenza. Ciascuna di questa due circostanze produce una diversa situazione di stallo. Come indicato nella diapositiva 48, il *deadlock* produce uno stallo con blocco: in tale situazione i *thread* coinvolti si trovano tutti in stato di attesa circolare per almeno una risorsa aggiuntiva, in presenza delle altre tre condizioni necessarie e sufficienti elencate in diapositiva 53. La *starvation*, invece, produce una situazione di stallo funzionale ma senza blocco (diapositiva 48): in tale situazione i *thread* coinvolti restano attivi senza però riuscire a far progredire le proprie attività. La diapositiva 50 mostra una soluzione erronea del problema dei filosofi a cena che produce *deadlock*, in quanto tutti i *thread* partecipanti acquisiscono risorse (sottraendole alla collettività) senza rilasciarle ove si rivelassero insufficienti al proprio scopo; quando tutti i *thread* coinvolti si comportassero allo stesso modo, essendo le risorse non prerilasciabili, si avrebbe il rischio di attesa circolare e dunque stallo con blocco. La diapositiva 51 mostra invece un'altra soluzione erronea dello stesso problema, questa volta esposta al rischio di *starvation*. In questo caso i *thread* incapaci di acquisire tutte le risorse necessarie rilasciano volontariamente quelle già acquisite con l'intento di provarci nuovamente in un tempo successivo. Tale scelta, pur generosa, non gode di garanzie di equità e pertanto qualche *thread* potrebbe indefinitamente trovarsi nella sitazione di doverci riprovare.

Soluzione 5 (punti 6). Il quesito richiede la descrizione dei principali passi di inizializzazione di un elaboratore equipaggiato con sistema operativo della famiglia UNIX/Linux, presumibilmente a partire dallo stato di "spento", dunque comprendendo l'intera sequenza gergalmente detta di *boot*. Gli elementi essenziali della risposta li troviamo alle diapositive 282-3 e 221-2 del corso dell'a.a. corrente. Ripercorriamo la sequenza:

1. all'accensione dell'elaboratore (assumendo si tratti di un PC di uso comune) va in esecuzione un programma residente in ROM, denominato BIOS (per *Basic Input Output System*, che fa il verso al greco antico $\beta\iota o\varsigma$, che significa "vita"); il BIOS verifica lo stato di salute dell'*hardware* di base dell'elaboratore secondo specifiche indicazioni di configurazione, e poi copia in RAM il contenuto dell'MBR (*Master Boot Record*) che si trova per convenzione nel settore 0 del disco di avvio, e lo manda in esecuzione

2. la prima (e maggior) parte dell'MBR contiene infatti un programma, anch'esso strettamente legato all'elaboratore, denominato *boot loader*, il cui compito è interpretare la tabella delle partizioni disponibili sul disco di avvio (e riportate nella parte finale dell'MBR) e avviare il caricamento di quella selezionata. Nella famiglia UNIX/Linux i *boot loader* più diffusi sono GRUB (ironicamente chiamato *GRand Unified Boot loader*) e LILO (più prosaicamente *LInux LOader*)

3. il *boot loader* non è ancora il sistema operativo e dunque non ne conosce il *file system*; esso deve però localizzare sul disco la partizione selezionata e da essa caricare in RAM il blocco di *boot*, che è specifico del corrispondente sistema operativo e che ha il compito di avviarne il caricamento e l'avviamento

4. il programma di *boot* del sistema operativo svolge un minimo di azioni di controllo e configurazione dell'*hardware* e poi avvia il *main* del sistema operativo. Da questo punto in poi, il codice di sistema operativo che viene caricato ed eseguito è prevalmentemente scritto in C e non più in linguaggio macchina perché sono finite le attività a basso livello strettamente legate all'*hardware* dell'elaboratore ospite. Tra i compiti principali dell'inizializzazione del sistema operativo vi è ovviamente la presa in carico del *file system*, ove risiedono i vari programmi di servizio e i loro valori di avviamento. Le informazioni che descrivono il *file system* e la posizione sul disco dei suoi dati si trovano in una struttura chiamata *superblocco.*

La seconda parte del quesito chiede di valutare la necessità di ogni singolo passo di tale sequenza, presumibilmente al fine di valutare l'eventualità di poter effettuare di qualche ottimizzazione. Vediamo: il passo 1 è ovviamente essenziale, perchè si svolge "a freddo", dunque senza potersi appoggiare su alcun altro programma preesistente; il passo 2 è ugualmente irrinunciabile perchè sarebbe troppo rigido e limitativo che fosse il BIOS stesso (normalmente scritto dal produttore dell'*hardware*, indipendentemente da quello del sistema operativo) a doversi occupare dell'MBR, che invece è intrinsecamente legato ai sistemi operativi installati dall'utente; il passo 3, che è ancora indipendente dal sistema operativo finale, è decisamente desiderabile in ambiente multi-partizione; eliminarlo significherebbe fissare a priori il numero di partizioni gestibili; il passo 4 infine è ovviamente necessario perché è solo da questo punto che il controllo passa effettivamente al sistema operativo desiderato.