Quesito 1 (punti 6). Cinque processi *batch*, identificati dalle lettere $A - E$ rispettivamente, arrivano all'elaboratore agli istanti $0, 3, 5, 7, 8$ rispettivamente. Tali processi hanno un tempo di esecuzione stimato di $5, 6, 3, 4, 2$ unità di tempo rispettivamente. Per ognuna delle seguenti politiche di ordinamento:

1. FPS (a priorità esplicita e costante, con prerilascio)
2. RR (a divisione di tempo, senza priorità e con quanto tempo di ampiezza 2)
3. SJF (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 *turn around*; (iii) il tempo medio di attesa.
Ove la politica di ordinamento in esame consideri i valori di priorità, tali valori, mantenuti staticamente per l'intera durata dell'esecuzione, sono rispettivamente: $4, 3, 5, 3, 5$ (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 2 (punti 7).
Dato il sistema descritto dalla seguente rappresentazione insiemistica delle assegnazioni di risorsa, dove $\mathcal{P}$ denota l'insieme dei processi, $\mathcal{R}$ l'insieme delle risorse R_i^j di indice i e molteplicità j, $\mathcal{E}(\mathcal{P})$ l'insieme delle richieste di accesso a risorse in $\mathcal{R}$ emesse da processi in $\mathcal{P}$ e *attualmente pendenti*, e $\mathcal{E}(\mathcal{R})$ l'insieme degli accessi *attualmente soddisfatti* di processi in $\mathcal{P}$ a risorse in $\mathcal{R}$:

$$\mathcal{P} = \{P_1; P_2; P_3; P_4; P_5; P_6\} \tag{1}$$
$$\mathcal{R} = \{R_1^1; R_2^2; R_3^2; R_4^1\} \tag{2}$$
$$\mathcal{E}(\mathcal{P}) = \{P_1 \to R_3; P_2 \to R_1; P_2 \to R_3; P_2 \to R_4; P_3 \to R_4; P_5 \to R_2; P_6 \to R_2\} \tag{3}$$
$$\mathcal{E}(\mathcal{R}) = \{R_1 \to P_1; R_2 \to (P_2, P_3); R_3 \to (P_4, P_5); R_4 \to P_6\} \tag{4}$$

si analizzi il grafo di allocazione delle risorse, determinando se il sistema i trovi attualmente in situazione di stallo o meno. Successivamente, si studi l'evoluzione dello stato del sistema ove il processo P_4 richiedesse accesso alla risorsa R_1 a partire dalla situazione data.

Quesito 3 (punti 7). Considerando come invariante la dimensione di un singolo settore su disco (come è noto fissata all'ampiezza di 512 B) si ipotizzino e si discutano concisamente le operazioni *logiche* con le quali un sistema operativo Unix "storico" arrivi a localizzare l'i-node di indice 42 relativo alla partizione di propria competenza. Si ricordi che per lo Unix storico la dimensione di un i-node è fissata a 64 B. Indicare poi:

1. almeno *due* delle ragioni che hanno spinto i progettisti di *file system* derivati da Unix di progettare un ampliamento dell'i-node;
2. un criterio guida per la scelta della dimensione di i-node più ampi per architetture a 32 *bit*.

Quesito 4 (punti 4).
[4.A]: Dato un sistema di *swapping* e una memoria con zone disponibili di ampiezza: $10, 4, 20, 18, 17, 9, 12, 15$ KB, in questo ordine, indicare quale area venga prescelta dalla politica next-fit a fronte della richiesta di caricamento di un segmento di ampiezza 3 KB dopo aver caricato un segmento ampio 12 KB:
1: 4 KB
2: 18 KB
3: 20 KB
4: 9 KB.

[4.B]: Data una partizione ampia 4 GB, con blocchi di ampiezza 4 KB, e contenente 128 K *file*, l'ampiezza della FAT dipende da:
1: il numero di *file* in essa rappresentati
2: l'ampiezza dei blocchi
3: l'ampiezza del disco in blocchi e l'ampiezza degli indici di blocco
4: l'ampiezza del disco.

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

[4.C]: Sia data memoria dotata di 4 *page frame*, inizialmente libere, e 8 pagine di memoria virtuale. Utilizzando la politica LRU per il rimpiazzo delle pagine, indicare quanti *page fault* si verifichino a fronte della stringa di riferimenti: 0172327103:
1: 4
2: 5
3: 6
4: 7.

[4.D]: In un confronto prestazionale tra *hard link* (HL) e *symbolic link* (SL):
1: gli HL sono da preferire perchè velocizzano gli accessi ai *file*
2: gli SL sono da preferire perchè assicurano la singolarità dell'associazione tra *directory* e *i-node*
3: i due sono sostanzialmente indistinguibili
4: gli SL hanno prestazioni superiori perchè impiegano meno spazio nella *directory*.

Quesito 5 (punti 8). Si propongano concisamente le linee essenziali di progettazione di un sistema di memoria virtuale paginata da realizzarsi sopra una architettura *hardware* a 32 *bit* che supporti segmentazione, ovvero l'associazione automatica tra un contesto di esecuzione e l'insieme dei segmenti che costituiscono il suo spazio di indirizzamento.
Alla luce del progetto proposto si mostri come in tale sistema avvenga la traduzione da indirizzo virtuale a indirizzo fisico.

Soluzione 1 (punti 6).

- FPS (a priorità esplicita e costante, con prerilascio)

```
processo A  AAAAA                   LEGENDA DEI SIMBOLI
processo B  ---bbbbbbbBBBBBB        - non ancora arrivato
processo C  -----CCC                x (minuscolo) attesa
processo D  -------ddddddddDDDD     X (maiuscolo) esecuzione
processo E  --------EE

CPU         AAAAACCCEEBBBBBBDDDD
coda        ...bbbbbbbddddd....
            .......ddd.........
```

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

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

```
processo A  AAAAaaA                 LEGENDA DEI SIMBOLI
processo B  ---bBBbbbBBbbbbbBB      - non ancora arrivato
processo C  -----ccCCccccccC        x (minuscolo) attesa
processo D  -------ddddDDddddddD    X (maiuscolo) esecuzione
processo E  --------eeeeeEE         . coda vuota

CPU         AAAABBACCBBDDEECBBDD
coda        ...baacbbddeeccbbbDD
            .....cbddeeccbbd....
            ........eccbbdd.....
```

processo	tempo di risposta	tempo di attesa	*turn-around*
A	0	2	2+5=7
B	1	1+3+5=9	9+6=15
C	2	2+6=8	8+3=11
D	4	4+5=9	9+4=13
E	5	5	5+2=7
medie	2,40	6,60	10,60

- SJF (senza considerazione di valori di priorità espliciti e con prerilascio)

```
processo A  AAAAA                   LEGENDA DEI SIMBOLI
processo B  ---bbbbbbbbbbbBBBBBB    - non ancora arrivato
processo C  -----CCC                x (minuscolo) attesa
processo D  -------dddDDDD          X (maiuscolo) esecuzione
processo E  --------EE

CPU         AAAAACCCEEDDDDBBBBBB
coda        ...bbbbddddbbbb......
            .......bbb.........
```

processo	tempo di risposta	tempo di attesa	tempo di *turn-around*
A	0	0	0+5=5
B	11	11	11+6=17
C	0	0	0+3=3
D	3	3	3+4=7
E	0	0	0+2=2
medie	2,80	2,80	6,80

Soluzione 2 (punti 7). La figura 1 riporta la versione grafica della rappresentazione insiemistica data.

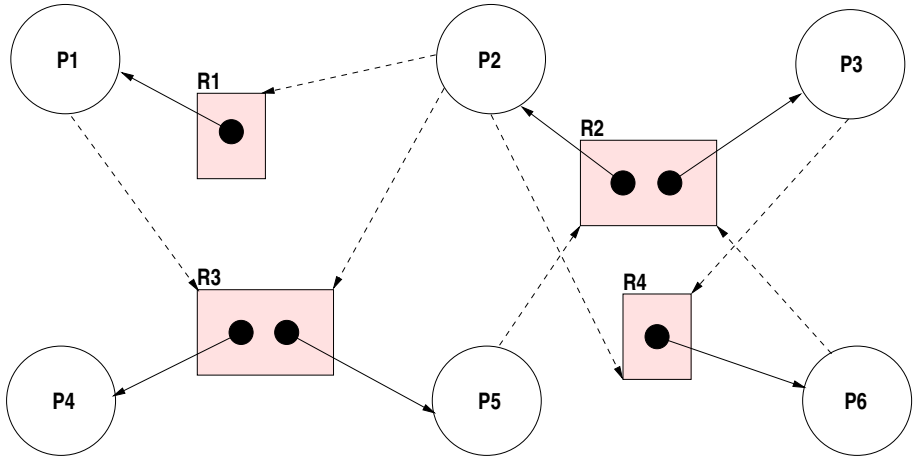

Figura 1: Grafo di allocazione delle risorse per il sistema dato, dove la freccia tratteggiata diretta da P_i a R_j denota una relazione in $\mathcal{E}(\mathcal{P})$ e la freccia solida diretta da R_j a P_i denota una relazione in $\mathcal{E}(\mathcal{R})$.

Allo stato si distinguono quattro percorsi chiusi:
percorso 1:

$$\{P_3 \rightarrow R_4 \rightarrow P_6 \rightarrow R_2 \rightarrow P_3\} \tag{5}$$

percorso 2:

$$\{P_2 \rightarrow R_3 \rightarrow P_5 \rightarrow R_2 \rightarrow P_2\} \tag{6}$$

percorso 3:

$$\{P_2 \rightarrow R_4 \rightarrow P_6 \rightarrow R_2 \rightarrow P_2\} \tag{7}$$

percorso 4:

$$\{P_1 \rightarrow R_3 \rightarrow P_5 \rightarrow R_2 \rightarrow P_2 \rightarrow R_1 \rightarrow P_1\} \tag{8}$$

entro i quali è presente almeno una risorsa completamente impegnata ma a molteplicità > 1 e quindi con almeno un processo potenzialmente non implicato nel percorso chiuso. In situazioni di questo genere *non si può* trarre alcuna conclusione a priori sullo stato di stallo del sistema, ma occorre analizzare il problema dato nel suo specifico dettaglio.
Tutti e 5 i percorsi condividino la risorsa R_2 a molteplicità > 1. Inoltre il percorso 2 e il percorso 4 condividono anche la risorsa R_3, anch'essa a molteplicità > 1. Questa situazione comporta la stessa caratteristica per tutti i percorsi dati: o sono tutti reciprocamente bloccati oppure tutti si potranno liberare successivamente.
Possiamo notare che una istanza della risorsa R_3 è assegnata al processo P_4 che non è sospeso in attesa di alcuna altra risorsa. Esso potrà pertanto procedere con la sua esecuzione fino a rilasciare la propria istanza della risorsa R_3. Quando ciò avverrà, due casi si potranno verificare in relazione a come verrà risolta la competizione di accesso ad essa da parte dei processi P_1 e P_2. Due situazioni sono possibili. Esaminiamole in dettaglio:

Caso 1 (Attribuzione di R_3 a P_1): il processo P_1 potrebbe a questo punto riprendere l'esecuzione e poi, al proprio completamento, rilasciare sia R_1 (totalmente) che R_3 (una singola istanza), entrambe le quali potrebbero essere assegnate al processo P_2, che però resterebbe in attesa di accesso alla risorsa R_4. In tal caso pertanto il sistema entrerebbe in uno stato di stallo coinvolgente i processi $\{P_2, P_3, P_5, P_6\}$.

Caso 2 (Attribuzione di R_3 a P_2): il processo P_2 non potrebbe comunque beneficiare di questa attribuzione, rimanendo in attesa delle risorse R_1 e R_4, il sistema trovandosi pertanto in uno stato di stallo che coinvolgerebbe i processi $\{P_1, P_2, P_3, P_5, P_6\}$.

Qualora infine il processo P_4 richiedesse accesso alla risorsa R_1 a partire dallo stato di sistema rappresentato in figura 1, esso entrerebbe inevitabilmente in una situazione di blocco definitivo (ovvero di stallo).

Soluzione 3 (punti 7). Come sappiamo, gli i-node di un *file system* Unix sono raccolti in una tabella posta in una zona prefissata della partizione, le cui coordinate (blocco di inizio, ampiezza, eventuale continuazione, etc.) sono reperibili nel superblocco. Essendo le strutture i-node indicizzate (ovvero invididualmente denotate da un indice numerico unico) ed essendo la loro tabella ordinata per indice, l'i-node di indice $N = 42$ e ampiezza $I = 64$ B, verrà banalmente localizzato determinando quale settore S_i di disco successivo alla base T della tabella contenga i dati ricercati, sapendo che i settori hanno ampiezza $S = 512$ B, ovvero:

$$S_i = T + (\lceil \tfrac{N \times I}{S} \rceil) - 1 = T + (\lceil \tfrac{42 \times 64}{512} \rceil) - 1 = T + 5.$$

In particolare, la base dell'i-node di indice $N = 42$ si troverà alla distanza 128 B (ovvero alla posizione logica 2, corrispondente alla III struttura i-node) del settore $T + 5$. È da notare che alla stessa conclusione saremmo arrivati considerando che un singolo settore di disco contiene 8 strutture i-node, per cui quella di indice 42 si troverà appunto nella posizione logica 2 del VI settore, che avrà indice 5.

Esempio di esigenze di ampliamento della struttura i-node:

1. crescente capienza dei dischi, con conseguente incremento nella cardinalità e dunque nell'ampiezza in *bit* degli indici di blocco
2. aumento delle informazioni e degli attributi di descrizione dei *file* rispetto a quelli basici e minimali previsti nello Unix "storico".

Criterio guida per la scelta della dimensione di i-node:

Precisamente per agevolare le operazioni logiche di localizzazione su disco è del tutto ovvio che la dimensione di un i-node dovrà sempre convenientemente essere un divisore o un multiplo dell'ampiezza di un settore e dunque necessariamente una potenza di 2. Non stupisce allora che il primo ampliamento ne abbia portato la dimensione a 128 B (come in Unix System V), poi successivamente estesa fino al valore tipico di 1 KB (come in molte incarnazioni di GNU/Linux).

Soluzione 4 (punti 4).

Quesito	Risposta
[4.A]	2
[4.B]	3 (ove per "disco" si intende "partizione")
[4.C]	4
[4.D]	1

Soluzione 5 (punti 8). Un sistema di memoria virtuale ibrido basato su "segmentazione paginata" può offrire vantaggi rilevanti rispetto alle scelte "pure", combinando insieme alcune delle caratteristiche interessanti dell'uno e dell'altro paradigma.
Tanto per cominciare, in un sistema del genere i segmenti dovranno avere indirizzo base allineato all'indirizzo di una pagina (contrariamente al caso puro in cui l'indirizzo può essere in qualunque posizione, con conseguente rischio di frammentazione esterna) e dimensione multipla di una pagina (tipicamente di ampiezza 4 KB), così che potranno essere portati in memoria secondo la classica forma di *paging on demand*, senza dunque requisiti di contiguità, con conseguente ottimizzazione dell'uso della memoria principale. A questi vantaggi derivanti dalla paginazione si aggiunge la maggiore capacità di controllo di accesso garantita dalla segmentazione che, disponendo di adeguato supporto di compilazione ed essendo disposti a sopportare il costo di frammentazione interna delle pagine, può essere spinta fino all'estremo della segmentazione pura che separa rigidamente per tipo le informazioni che costituiscono la memoria virtuale.
Il supporto *hardware* alla segmentazione velocizza l'associazione tra un processo (la sua memoria virtuale) e l'insieme dei segmenti in cui essa è strutturata, tipicamente mettendo a disposizione registri veloci in cui caricare informazioni chiave per la traduzione degli indirizzi virtuali in indirizzi fisici.

In tale sistema ogni processo è rappresentato da una tabella di segmenti (nel caso del Pentium capace di indirizzare fino a 8 K segmenti), la quale potrà ovviamente essere essa stessa paginata. La tabella dei segmenti di un processo è immediatamente localizzabile a partire da un corrispondente descrittore nella tabella dei processi. Ad ogni istante di esecuzione esisterà un singolo segmento corrente, denotato da un indice nella suddetta tabella.
Un segmento è caratterizzato da una base, che denota un indirizzo in memoria principale (espresso su 32 *bit* nel caso di architetture con tale unità) e un limite espresso in pagine, in virtù della paginazione interna dei segmenti (e dunque necessariamente limitato a 20 *bit* in architetture a 32 *bit*, per denotare fino a 1 M pagine di ampiezza 4 KB, ovvero uno spazio ampio fino a 4 GB per singolo segmento.
Con questa premessa possiamo comprendere come un indirizzo virtuale in tale sistema possa essere tradotto in un indirizzo fisico. La figura 2 descrive gli elementi di tale traduzione.

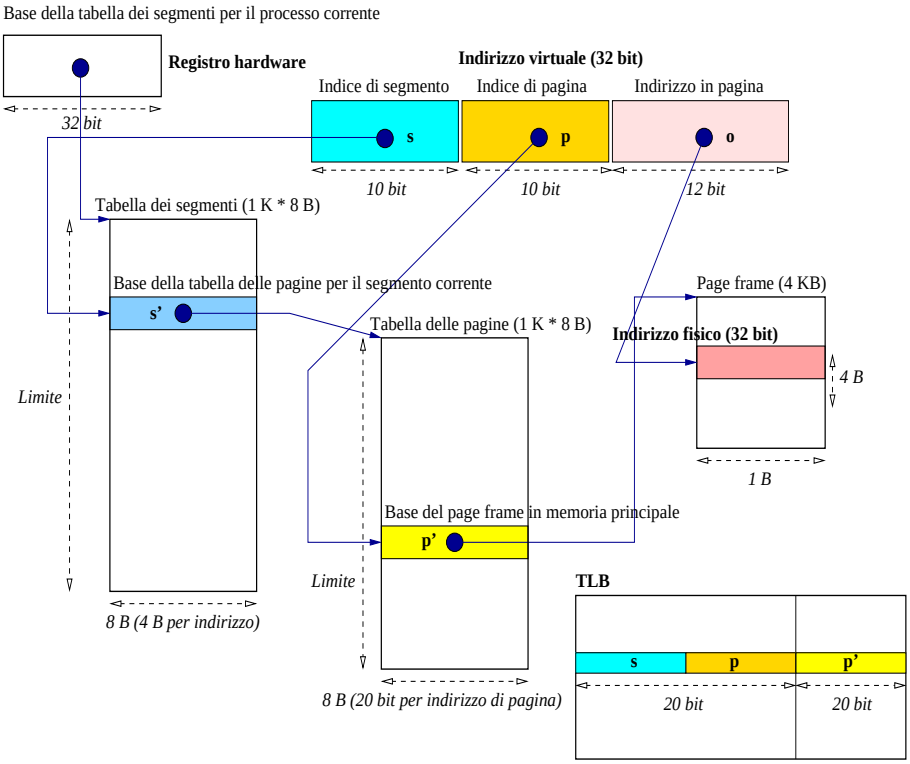

Figura 2: Traduzione di indirizzi virtuali in indirizzi fisici in un sistema basato su segmentazione paginata per un'architettura a 32 *bit*.

È evidente come per evitare un decadimento inaccettabile delle prestazioni conseguente ai tre cicli di memoria necessari per effettuare la traduzione degli indirizzi illustrata in figura 2 diventi fondamentale l'uso di supporto *hardware* di accelerazione. A questo scopo si utilizza tipicamente memoria associativa nella forma di strutture dette TLB (Translation Look-aside Buffer) che contengono alcune delle corrispondenze mostrate in figura e che vengono gestite con politiche di rimpiazzo simili a quelle utilizzate per la *cache*.