Esercizi per il recupero

1. convertire 25493_{10} a binario (16 bit) e ad esadecimale; bisogna dividere per due considerando quoziente e resto

	25493 | 1	← resto
quoziente	→ 12746 | 0	
	6373 | 1	
	3186 | 0	
	1593 | 1	
	796 | 0	
	398 | 0	
	199 | 1	
	99 | 1	
	49 | 1	
	24 | 0	
	12 | 0	
	6 | 0	
	3 | 1	
	1 | 1	↑

$25493_{10} = 110001110010101_2 = 0110.0011.1001.0101_2 = 6395_{16}$

Esercizio 2 - Conversioni

Effettuare le seguenti conversioni, tenendo conto del numero di bit con cui si rappresenta il numero da convertire/convertito.

1. 12 bit $(7B9)_H = (?)_2 = (?)_8 = (?)_{10}$ (unsigned/M+S/complemento a 2)
2. 11 bit $(7B9)_H = (?)_{10}$ (segno + modulo) $= (?)_{10}$ (complemento a 2)

Esercizio 3 – svolgere le richieste seguenti

3. convertire 12.6_{10} a binario (16 bit) in virgola fissa (8 bit per la parte intera, 8 per quella frazionaria);

   $12,6_{10} = 12 + 0.6$

   conversione in binario della parte intera rappresentata con 8 bit $12_{10} = 00001100_2$

   conversione in binario della parte frazionaria con 8 bit $0.6_{10} = 10011001_2$ (parte frazionaria periodica)

   in definitiva $12,6_{10} = 00001100.10011001_2$
4. convertire 56_{10} a binario (prima a 8 bit e poi a 16 bit);

   $56_{10} = 00111000_2$ con otto bit $= 0000000000111000_2$ con sedici bit
5. convertire -56_{10} a binario in complemento a 2 (prima a 8 bit e poi a 16 bit);

   la rappresentazione con 8 bit di $+56_{10}$ è 00111000_2 dove il bit più a sinistra rappresenta il segno +

   per convertire -56_{10} basta complementare tutti i bit della rappresentazione di $+56_{10}$ ed aggiungere 1 al bit meno significativo cioè $11000111_2 +$

   1_2

   11001000_2

   $+56_{10}$ con 16 bit: 0000000000111000 dove il bit più a sinistra rappresenta il segno +

   -56_{10} con 16 bit: 1111111111001000 dove il bit più a sinistra rappresenta il segno +

Esercizio 4 - Floating point

1. Si rappresenti in formato IEEE 754 single precision il valore $(56{,}25)_{10}$
2. Dato il seguente valore rappresentato in IEEE 754 single precision, si determini il valore in decimale con notazione normalizzata $x,y \times 10^z$. **97180000**.

6. constatare che il complemento a 2 di -56_{10} (binario a 8 bit) fornisce $+56_{10}$ (binario a 8 bit);

00110111_2 + (complementazione dei bit che rappresentano -56_{10})

1_2

00111000_2 cioè $+56_{10}$

7. trovare la codifica floating-point su 32 bit (standard IEEE 754) di 12.6_{10} ed esprimerla in notazione esadecimale;

$12{,}6_{10} = 1100{,}1001_2$

bisogna normalizzare la mantissa $1{,}1001001_2 * 2^3$ quindi E=3

segno + quindi il bit 31 della codifica deve essere posto a 0

codifica degli 8 bit relativi all' esponente e :

e = E +127 quindi e = 3+127 =130 in binario 10000010_2

per quando riguarda la mantissa bisogna ricordare che essa è periodica per cui nei 23 bit riservati si inserisce, partendo da sinistra verso destra, la parte non periodica della mantissa poi la parte periodica quest'ultima replicata fino al riempimento di tutti i bit a disposizione quindi

10010011001100110011001

in definitiva 01000001010010011001100110011001

in esadecimale 0100.0001.0100.1001.1001.1001.1001.1001

4 1 4 9 9 9 9 9

41499999_{16}

Esercizio 5

Effettuare i seguenti cambiamenti di codifica su numeri naturali:

- $123_{10} = x_2$ [1111011_2]
- $011101_2 = x_{10}$ [29_{10}]
- $23_{10} = x_5$ [43_5]
- $123_5 = x_{10}$ [38_{10}]
- $123_{10} = x_H$ [$7B_{16}$]
- $A1_H = x_{10}$ [161_{10}]
- $91_H = x_Q$ [221_8]
- $123_{10} = x_{BCD}$ [000100100011_{BCD}]
- $10010110_{BCD} = x_{10}$ [96_{10}]
- $10100110_{BCD} = x_{10}$ [impossibile: 1010 non è una cifra valida in BCD]
- $10010110_{BCD} = x_2$ [1100000_2]

Esercizio 6

Effettuare i seguenti cambiamenti di codifica su numeri relativi considerando numeri binari da 6 bit:

- $+13_{10} = x_{MS}$ [001101_{MS}]
- $-13_{10} = x_{MS}$ [101101_{MS}]
- $+32_{10} = x_{MS}$ [non rappresentabile in codifica MS su 6 bit]
- $-32_{10} = x_{MS}$ [non rappresentabile in codifica MS su 6 bit]
- $+13_{10} = x_{CA2}$ [001101_{CA2}]
- $-13_{10} = x_{CA2}$ [110011_{CA2}]
- $+32_{10} = x_{CA2}$ [non rappresentabile in codifica CA2 su 6 bit]
- $-32_{10} = x_{CA2}$ [100000_{CA2}]
- $010001_{MS} = x_{10}$ [$+17_{10}$]
- $010001_{CA2} = x_{10}$ [$+17_{10}$]
- $100001_{MS} = x_{10}$ [-1_{10}]
- $100001_{CA2} = x_{10}$ [-31_{10}]

Esercizio 7

Effettuare le seguenti operazioni considerando numeri binari da 6 bit ed indicando sempre se si è verificato errore e di quale tipo:

- (binario puro) `010101 + 000111` [011100, ok]
- (binario puro) `010101 + 010001` [100110, ok]
- (binario puro) `010101 - 000111` [001110, ok]
- (binario puro) `010101 - 011001` [111100, overflow: borrow su MSB]
- (CA2) `010101 + 000111` [011100, ok]
- (CA2) `010101 + 010001` [100110, overflow: cambio di segno]
- (CA2) `010101 - 000111` [001110, ok]
- (CA2) `010101 - 011001` [111100, ok]
- (binario puro) $010101 \ll 1$ [101010, ok]
- (binario puro) $010101 \ll 2$ [010100, overflow: scarto di un bit a 1]
- (binario puro) $110101 \gg 1$ [011010, troncamento: scarto di un bit a 1]

Esercizio 8

Effettuare la seguenti conversioni mantenendo la stessa precisione assoluta:

- $10.011_2 = x_{10}$ [2_{10}]
- $10.0110_2 = x_{10}$ [2.3_{10}]
- $0.0101001000_2 = x_{10}$ [0.320_{10}]

Esercizio 9

Convertire in formato IEEE-754 SP i seguenti numeri decimali:

- -7 [1 10000001 11000000000000000000000]
- $+3.54 \cdot 10^{-3}$ [0 01110110 11001111111111101011000]

Esercizio 10

Convertire in decimale i seguenti numeri binari in formato IEEE-754 SP, esprimendo il risultato in forma esponenziale ingegneristica con la stessa precisione del numero binario:

- `01101100001010000000000000000000` $[+812.3981 \cdot 10^{24}]$

  Il MSB rappresenta il segno: il numero è quindi positivo. Gli 8 bit successivi al MSB rappresentano l'esponente (in codifica "eccesso 127"):

  $$11011000_2 - 127_{10} = 216 - 127 = 89$$

  I restanti 23 bit rappresentano la parte frazionaria del modulo della mantissa (che ha la parte intera "1." sottintesa), quindi il numero codificato è:

  $$\begin{aligned} +1.0101_2 \cdot 2^{89} &= +10101_2 \cdot 2^{85} \\ &= +21 \cdot 2^{85} \\ &= +812,398,150,781,030,805,402,550,272 \\ &= +812.3981 \cdot 10^{24} \end{aligned}$$

  Si noti che, trattandosi di un numero in codifica IEEE 754 SP, il risultato è stato scritto con 7 cifre significative. Inoltre, essendo stata richiesta la rappresentazione in formato esponenziale ingegneristico, si è scelta come esponente decimale il massimo multiplo di tre che permette di avere una parte intera non nulla.

- `10000111100000000000000000000000` $[-192.5930 \cdot 10^{-36}]$

  Il MSB rappresenta il segno: il numero è quindi negativo. Gli 8 bit successivi al MSB rappresentano l'esponente (in codifica "eccesso 127"):

  $$00001111_2 - 127_{10} = 15 - 127 = -112$$

  I restanti 23 bit rappresentano la parte frazionaria del modulo della mantissa (che ha la parte intera "1." sottintesa), quindi il numero codificato è:

  $$\begin{aligned} -1.0_2 \cdot 2^{-112} &= -2^{-112} \\ &= -1.925929944 \cdot 10^{-34} \\ &= -192.5930 \cdot 10^{-36} \end{aligned}$$

  Si noti che, trattandosi di un numero in codifica IEEE 754 SP, il risultato è stato scritto con 7 cifre significative. Inoltre, essendo stata richiesta la rappresentazione in formato esponenziale ingegneristico, si è scelta come esponente decimale il massimo multiplo di tre che permette di avere una parte intera non nulla.

Sistemi operativi

Esercizio 1 – Scheduling 1

In un sistema operativo che adotta uno scheduling con diritto di prelazione, quattro processi arrivano al tempo indicato e consumano la quantità di CPU indicata nella tabella sottostante)

Processo	T. di arrivo	Burst
P1	0	13
P2	3	8
P3	6	4
P4	8	1

a)
Qual è il waiting time medio migliore (ossia ottimale) che potrebbe essere ottenuto per lo scheduling dei quattro processi della tabella? RIPORTATE IL DIAGRAMMA DI GANTT USATO PER IL CALCOLO. (lasciate pure i risultati numerici sotto forma di frazione, e indicate quali assunzioni fate)

Diagramma di GANT, assumendo come algoritmo di scheduling SJF preemptive:
(0)….P1 …(3) …. P2….(6)….P3….(8)….P4….(9)….P3…(11)….P2…(16)…P1….(26)

Waiting time medio:
P1 = (26 – 0) – 13 = 13;
P2 = (16 – 3) – 8 = 5;
P3 = (11 – 6) – 4 = 1;
P4 = (9 – 8) – 1 = 0;
waiting time medio = 19/4

Esercizio – Scheduling 2

Considerate il seguente insiemi di processi, con la durata della sequenza di operazioni della CPU espressa in millisecondi:

Processo	Durata della sequenza	Priorità
P_1	3	1
P_2	3	2
P_3	5	3
P_4	6	2
P_5	7	1

Supponendo che i processi arrivino nell'ordine P_1, P_2, P_3, P_4, P_5, e siano tutti presenti al tempo 0.

1. Disegnare quattro diagrammi di Gantt per illustrare l'esecuzione di questi processi con gli algoritmi di scheduling 1)FCFS, 2)SJF, 3)con priorità senza prelazione (un valore di priorità più basso indica una priorità maggiore) e 4)RR (quanto=2).
2. Calcolare il tempo di completamento di ciascun processo per ogni algoritmo di scheduling.
3. Calcolare il tempo di attesa di ciascun processo per ogni algoritmo di scheduling.
4. Per quale algoritmo in questo caso l'esecuzione di ogni processo ha in media il minimo tempo d'attesa?

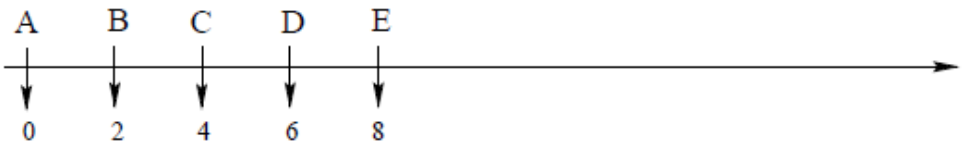
Esercizio – Scheduling 3

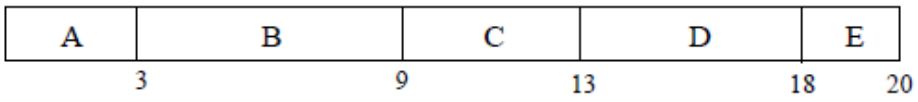
Si consideri il seguente insieme di processi:

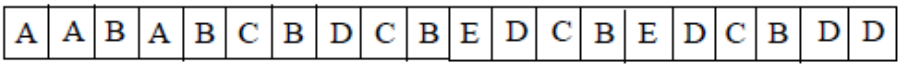
processo	*tempo di arrivo*	*CPU–burst (millisec.)*
A	0	3
B	2	6
C	4	4
D	6	5
E	8	2

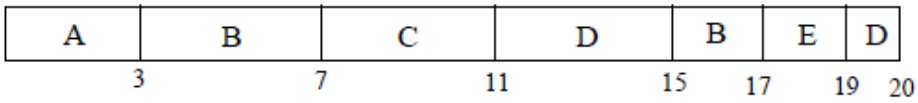
Si calcoli il tempo medio di attesa ed il tempo medio di turnaround, nel caso di scheduling FCFS, RR con quanti di tempo 1 e 4 e SJF non preemptive.

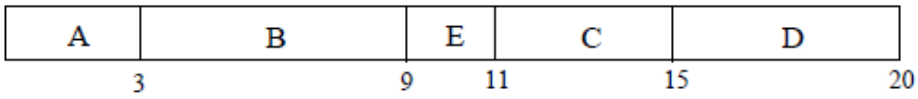
Soluzione











$$t_a(FCFS) = \frac{(3-3-0)+(9-6-2)+(13-4-4)+(18-5-6)+(20-2-8)}{5} = 4.6$$

$$t_{tr}(FCFS) = \frac{(3-0)+(9-2)+(13-4)+(18-6)+(20-8)}{5} = 8.6$$

$$t_a(RR-1) = \frac{(4-3-0)+(18-6-2)+(17-4-4)+(20-5-6)+(15-2-8)}{5} = 6.8$$

$$t_{tr}(RR-1) = \frac{(4-0)+(18-2)+(17-4)+(20-6)+(15-8)}{5} = 10.8$$

$$t_a(RR-4) = \frac{(3-3-0)+(17-6-2)+(11-4-4)+(20-5-6)+(19-2-8)}{5} = 6$$

$$t_{tr}(RR-4) = \frac{(3-0)+(17-2)+(11-4)+(20-6)+(19-8)}{5} = 10$$

$$t_a(SJF) = \frac{(3-3-0)+(9-6-2)+(15-4-4)+(20-5-6)+(11-2-8)}{5} = 3.6$$

$$t_{tr}(SJF) = \frac{(3-0)+(9-2)+(15-4)+(20-6)+(11-8)}{5} = 7.6$$

Esercizio – Scheduling 4

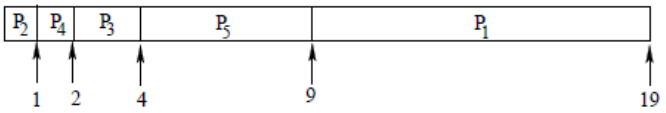
Si considerino cinque processi caratterizzati dai seguenti tempi di esecuzione (in millisecondi) e priorità date esternamente (un codice di priorità più piccolo indica una priorità più alta):

Processo no.	*Tempo di esecuzione*	*Priorità*
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

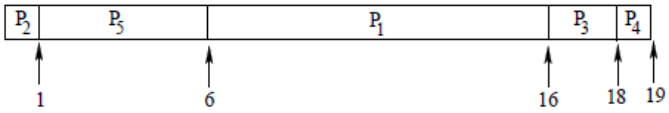
I processi usano solo la CPU ed arrivano tutti al tempo 0 nell'ordine P_1, P_2, P_3, P_4, P_5. Si illustri quale risulta l'ordine di esecuzione nel caso delle politiche SJF ed a priorità. Si calcoli il tempo di attesa medio nei due casi.

Soluzione

SJF



Scheduling a Priorita'



$$t_a(SJF) = \frac{9+0+2+1+4}{5} = 3.2$$

$$t_a(Pr) = \frac{6+0+16+18+1}{5} = 8.2$$

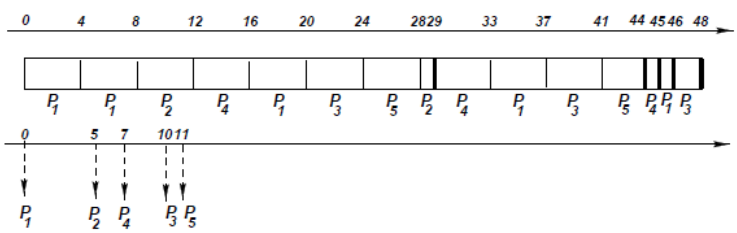
Esercizio – Scheduling 5

Si consideri un insieme di cinque processi P_1, P_2, P_3, P_4, P_5 con i seguenti tempi di arrivo e tempi di esecuzione in millisecondi:

processo	*tempo di arrivo*	*tempo di esecuzione*
P_1	0	17
P_2	5	5
P_3	10	10
P_4	7	9
P_5	11	7

Assegnare l'insieme di processi ad un processore in base alla politica Round Robin considerando un quanto di tempo di 4 millisecondi. Calcolare il valor medio del tempo di attesa ed il valor medio del tempo di turnaround dei processi.

Soluzione



$$t_a = \frac{29+19+28+29+26}{5} = 26.2 \text{ msec}$$

$$t_{tr} = \frac{46+24+38+38+33}{5} = 35.8 \text{ msec}$$

Esercizio – Memoria 1

Si consideri un sistema in cui in una tabella delle pagine di un processo l'indice più grande usabile nella tabella delle pagine di quel processo può essere 7FFF. Un indirizzo fisico del sistema è scritto su 25 bit, e la RAM è suddivisa in 4000 (esadecimale) frame.

a)
Quanto è grande, in megabyte, lo spazio di indirizzamento logico del sistema (esplicitate i calcoli che fate)?

4000(esadecimale) = 2^{14}, per cui un numero di frame è scritto su 14 bit, e la dimensione di un frame, e quindi di una pagina, è di $2^{\wedge 11}$ byte (25 - 14 = 11). Poiché il numero più grande di una pagina è 7FFF, ci possono essere al massimo $2^{\wedge 15}$ pagine, e lo spazio di indirizzamento logico è di $2^{\wedge 15}$ x $2^{\wedge 11}$ byte (pari a circa 64 megabyte).

b)
Indicate tutte e sole le informazioni contenute in una entry di una tabella delle pagine di questo sistema, se il sistema usa l'algoritmo di rimpiazzamento della seconda chance.

Il numero del frame che contiene la pagina corrispondente, il bit di validità della pagina, il reference bit.

Esercizio – Memoria 2

Un sistema di allocazione della memoria ha le seguenti pagine libere, in questo ordine:
8KB, 15KB, 3KB, 11KB, 5KB, 7KB, 20KB, 25KB.
Si considerino tre richieste di allocazione che arrivano, una di seguito all'altra, nel seguente ordine:
A) 11K;
B) 4K;
C) 13K.

Indicare a quali pagine vengono assegnate le tre richieste sequenziali A, B e C considerando le politiche *First Fit, Next Fit, Best Fit* e *Worst Fit*.
Nota: si assuma che, qualora un blocco libero venga assegnato a seguito di una richiesta di dimensione inferiore, il blocco libero sia comunque interamente assegnato.

	A)	B)	C)
First Fit			
Next Fit			
Best Fit			
Worst Fit			

Esercizio –Memoria 3

In un sistema in cui sono disponibili tre frame dire quante assenze di pagine avvengono per la seguente successione di riferimenti usando gli algoritmi di sostituzione secondo l'ordine di arrivo(FIFO), ottimale e usate meno recentemente (LRU).

1,2,3,3,2,1,4,5,3,6,1,2,3,3,2,5

Esercizio – Memoria 4

Si consideri un sistema in cui in una tabella delle pagine di un processo l'indice più grande usabile nella tabella delle pagine di quel processo può essere 3FFF. Un indirizzo fisico del sistema è scritto su 24 bit, e la RAM è suddivisa in 2000 (esadecimale) frame.

a)
Quanto è grande, in megabyte, lo spazio di indirizzamento logico del sistema (esplicitate i calcoli che fate)?

2000(esadecimale) = 2^{13}, per cui un numero di frame è scritto su 13 bit, e la dimensione di un frame, e quindi di una pagina, è di 2^{11} byte (24 - 13 = 11). Poiché il numero più grande di una pagina è 3FFF, ci possono essere al massimo 2^{14} pagine, e lo spazio di indirizzamento logico è di 2^{14} x 2^{11} byte (pari a circa 32 megabyte).

b)
Quali informazioni conterrà ciascuna entry di una tabella delle pagine di questo sistema, se il sistema usa l'algoritmo di rimpiazzamento della seconda chance?

Il numero del frame che contiene la pagina corrispondente, il bit di validità della pagina, il reference bit.

Esercizio – Memoria 5

Si consideri un processo che fa rifermento a 5 pagine virtuali nel seguente ordine:

$$1, 2, 2, 3, 2, 3, 4, 2, 2, 3, 5, 1, 3, 5$$

Si consideri una memoria fisica (inizialmente vuota) di 3 frame e si mostri il funzionamento degli algoritmi seguenti:

1. FIFO (First In First Out),

Spiegare brevemente:

	1	2	2	3	2	3	4	2	2	3	5	1	3	5
Frame 1														
Frame 2														
Frame 3														
Page Fault														

2. LRU (Least Recently Used),

Spiegare brevemente:

	1	2	2	3	2	3	4	2	2	3	5	1	3	5
Frame 1														
Frame 2														
Frame 3														
Page Fault														

3. LFU (Least Frequently Used).

Spiegare brevemente:

	1	2	2	3	2	3	4	2	2	3	5	1	3	5
Frame 1														
Frame 2														
Frame 3														
Page Fault														