

**PROVA SCRITTA DEL CORSO DI**  
**CALCOLATORI ELETTRONICI**  
**NUOVO E VECCHIO ORDINAMENTO DIDATTICO**  
24 Settembre 2008

**NOME:**

**COGNOME:**

**MATRICOLA:**

**ESERCIZIO 1 (NO: 8 punti - VO: 7 punti)**

Si vogliano sommare i seguenti quattro numeri a 8 bit:

$X1 = 00100111$ ,  $X2 = 10101101$ ,  $X3 = 01100001$ ,  $X4 = 11010010$ .

1. (NO: 5 punti, VO: 4 punti) Disegnare lo schema che permette di eseguire tale somma usando due addizionatori del tipo "Carry Save Adder" ed un "Parallel Adder" finale. Gli addizionatori "Carry Save Adder" lavorano su tre operandi. Precisare il valore assunto dalla pseudosomma e dal pseudoriporto all'uscita del primo e del secondo "Carry Save Adder" e le operazioni eseguite per ottenere la somma finale.
2. (3 punti) Nell'ipotesi che  $d$  sia il ritardo per il calcolo di somma e riporto per un full-adder, calcolare il ritardo del presente sistema, indicando il vantaggio che deriva rispetto all'uso di soli parallel adder.

**ESERCIZIO 2 (NO: 9 punti – VO: 8 punti)**

Si consideri un calcolatore che dispone di una memoria principale di 64 kbyte suddivisa in blocchi di 8 byte. E' possibile accedere al singolo byte e la modalità di indirizzamento usata per la cache, costituita da 32 blocchi indirizzabili, sia quella "diretta".

1. (NO: 2 punti – VO: 1 punto) Spiegare, precisando il significato e la funzione dei diversi campi, come vengono interpretati gli indirizzi logici per recuperare l'informazione contenuta nella cache.
2. (2 punti) Indicare in quali blocchi di primaria si trovano i seguenti byte (indirizzi in esadecimale): 113B, C334, 1138, AAAA.
3. (3 punti) Indicare in quali blocchi di cache devono essere memorizzati i byte del passo precedente. Se tali parole venissero richieste sequenzialmente, quanti sarebbero gli hit di cache (ipotizzando la cache inizialmente vuota) ?
4. (2 punti) Si supponga che il byte di indirizzo 1A1B sia memorizzato in cache. Indicare gli indirizzi di tutti gli altri byte memorizzati nello stesso blocco di cache.

**ESERCIZIO 3 (solo VO: 7 punti)**

Si consideri il caso di un Bus di I/O sul quale devono essere collegate 8 periferiche esterne.

1. (4 punti) Si ipotizzi che sulla parte di controllo del Bus si abbiano solo 2 segnali di controllo liberi da poter utilizzare per gestire l'arbitraggio delle 8 periferiche da collegare. Descrivere la tecnica di arbitraggio utilizzabile in una tale situazione, disegnando un possibile schema di collegamento delle periferiche e spiegando chiaramente le implicazioni del suo utilizzo sulla gestione delle richieste di I/O delle periferiche.
2. (3 punti) Nelle ipotesi di cui al punto precedente, sarebbe utilizzabile la tecnica di arbitraggio centralizzato con richieste indipendenti per ciascuna periferica ? In caso di risposta negativa, cosa sarebbe necessario modificare per renderla utilizzabile ?

**ESERCIZIO 3 (solo NO: 8 punti)**

Implementare in Assembly MIPS una funzione che, dati l'indirizzo iniziale di un vettore  $v$  (in \$4) e due indici  $i$  e  $j$  (rispettivamente in \$5 e in \$6), scambi  $v(i)$  con  $v(j)$  solo se  $v(i) <> v(j)$  (ovvero solo se  $v(i)$  è diverso da  $v(j)$ ).

Vincolo: si usi il minimo numero di registri, senza preoccuparsi di preservare i valori contenuti nei registri destinati ai parametri della funzione eccetto che per il contenuto di \$4.

**ESERCIZIO 4 (NO: 8 punti - VO: 6 punti)**

Sia data la seguente lista di processi (si supponga che l'istante iniziale sia 0):

Job	Tempo di Arrivo	Tempo di CPU Richiesto	Richiesta di Memoria
1	0.0	2.0	20K
2	0.5	1.0	40K
3	1.0	1.0	50K
4	1.5	0.25	60K
5	2.0	1.0	20K

Il sistema ha 100K di memoria disponibile, e viene gestita con partizioni dinamiche non rilocabili.

1. (NO: 5 punti - VO: 4 punti) Mostrare, utilizzando il metodo grafico, la sequenza di esecuzione dei job qualora si impieghi la politica di scheduling FIFO multiprogrammata "round robin".
2. (NO: 3 punti - VO: 2 punti) Indicare, negli istanti in cui un job va in esecuzione e termina, la partizioni di memoria occupate da ciascun job e le partizioni libere, giustificando l'eventuale "messa in attesa" dei job.

**ESERCIZIO 5 (solo VO: 5 punti)**

Spiegare in modo chiaro e sintetico le differenze fra un'architettura CISC e un'architettura RISC.

**ESERCIZIO 1****Soluzione.**

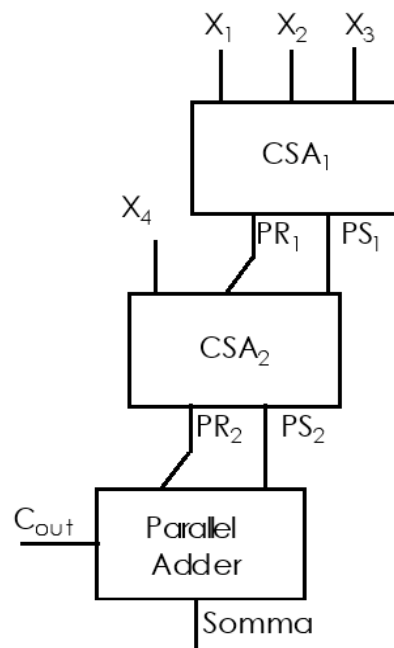
1.

$$PS_i = A_i \oplus B_i \oplus C_i$$

$$PR_i = A_i B_i + A_i C_i + B_i C_i$$

$$S = PS + 2PR$$

Schema:



X1	00100111	PS1	11101011	PS2	01110011	
X2	10101101	2PR1	01001010	2PR2	10010101	
X3	01100001	X4	11010010			
PS1	11101011	PS2	01110011	SUM		00000111
PR1	00100101	PR2	11001010	CARRY		10 (overflow)

2. Il tempo di ritardo del sistema con l'uso di tre Parallel Adder è  $8d+8d+8d=24d$ , mentre con il Carry Save Adder equivale a  $8d+d+d=10d$ .

## ESERCIZIO 2

### Soluzione

(a) <Tag 8 bit> <Cache Index 5 bit> <Offset 3 bit>

(b-c)

113B → 0001 0001 | 0011 1 | 011 → Block frame (551)<sub>10</sub> → Cache index 7  
C334 → 1100 0011 | 0011 0 | 100 → Block frame (6246)<sub>10</sub> → Cache index 6  
1138 → 0001 0001 | 0011 1 | 000 → Block frame (551)<sub>10</sub> → Cache index 7  
AAAA → 1010 1010 | 1010 1 | 010 → Block frame (5461)<sub>10</sub> → Cache index 21

Se le parole venissero richieste sequenzialmente si verificherebbe un hit (la terza parola è già stata memorizzata in cache dopo il primo accesso).

(d)

Essendo:

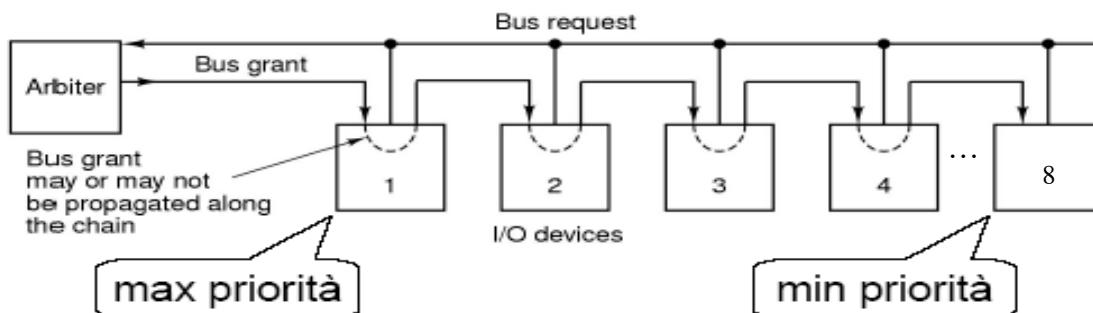
1A1B → 0001 1010 0001 1 | 011,

si ottiene facilmente che gli altri byte contenuti nello stesso blocco sono: 1A18 (offset 000), 1A19 (offset 001), 1A1A (offset 010), 1A1C (offset 100), 1A1D (offset 101), 1A1E (offset 110), 1A1F (offset 111).

### ESERCIZIO 3 (solo VO)

#### Soluzione.

1) Con soli 2 segnali liberi è possibile solo utilizzare la tecnica "daisy chain" che ha solo 2 segnali (request e grant). Ovviamente questa tecnica implica una gestione rigida delle priorità, che sono definite una volta per tutte.



2) Non è ovviamente possibile. Bisognerebbe avere almeno  $8 \times 2 = 16$  segnali di controllo liberi.

### ESERCIZIO 3 (solo NO)

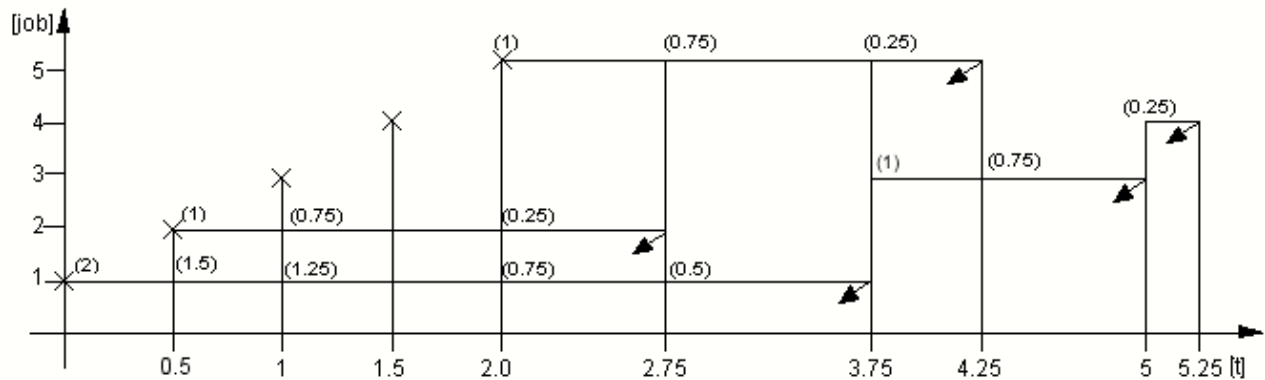
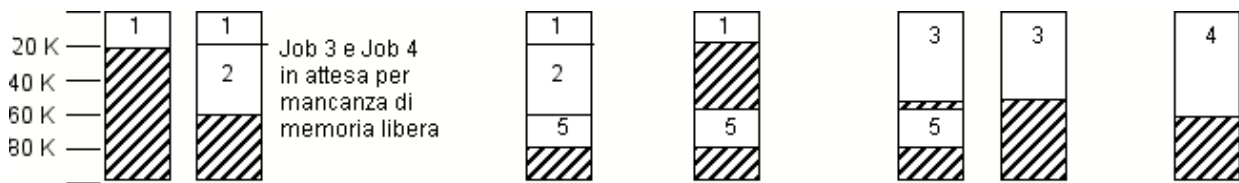
#### Soluzione.

```
$9 ← v[i]; $10 ← v[j]
```

```
funzione: addi $29, $29, -8
          sw $9, 0($29)
          sw $10, 4($29)
          muli $5, $5, 4
          add $5, $5, $4      #indirizzo di v[i] in $5
          muli $6, $6, 4
          add $6, $6, $4      #indirizzo di v[j] in $6
          lw $9, 0($5)
          lw $10, 0($6)
          beq $9, $10, exit   #if v[i]==v[j] exit
          sw $9, 0($6)
          sw $10, 0($5)
exit:     lw $9, 0($29)
          lw $10, 4($29)
          addi $29, $29, 8
          jr $31
```

## ESERCIZIO 4

### Soluzione



Job	$t_{\text{arrivo}}$	$t_{\text{start}}$	$t_{\text{finish}}$	Turnaround time	Weighted Turnaround time
1	0	0	3.75	3.75	1.875
2	0.5	0.5	2.75	2.25	2.25
3	1	3.75	5	4	4
4	1.5	5	5.25	3.75	15
5	2	2	4.25	2.25	2.25
<b>Media</b>				<b>3.2</b>	<b>5.075</b>

## ESERCIZIO 5 (solo VO)

### Soluzione

Si vedano le dispense del corso.