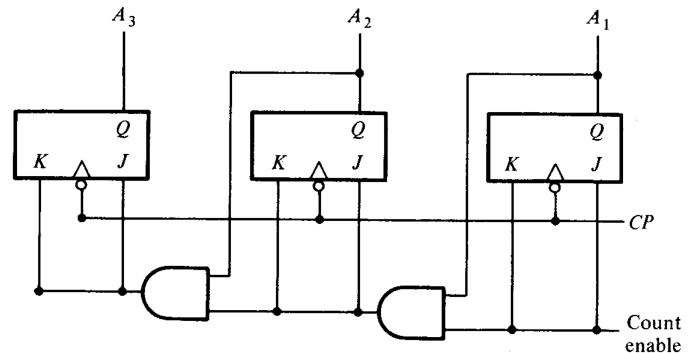


08 Giugno 2010

**MATRICOLA:**

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. (2 punti) 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, 1137, AAAA. Non è necessario convertire i valori dei blockframe in decimale.
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) ?

Si scriva una funzione Assembly MIPS, chiamata `media_k` che, ricevendo in ingresso l'indirizzo iniziale di un vettore in `$4` di dimensione `N` (in `$5`), ed un intero `k < N` (in `$6`), metta in `$7` il valore  $(v[k] + v[k+1]) / 2$ . Se  $k = N - 1$ , metta in `$7` il valore  $(v[N-1] + v[0]) / 2$ . Si faccia uso di una funzione esistente `media` che, ricevendo due interi in `$4` e `$5`, mette in `$6` il loro valor medio. Si abbia cura di non avere alterato i contenuti di `$4`, `$5`, `$6` all'uscita dalla funzione `media_k`.

Si considerino i numeri interi decimali 193 e 104.

1. (2 punti) Rappresentare i due numeri nel formato binario in virgola mobile IEEE 754 (32 bit, con mantissa frazionaria e normalizzata rappresentata in segno e valore e esponente a 8 bit in eccesso 127).
2. (3 punti) Sommare i due numeri rappresentati in virgola mobile seguendo i passi dell'algoritmo usato nei calcolatori (*senza dimenticare il bit implicito!*).
3. (2 punti) Descrivere gli elementi di una ALU che effettua somme fra numeri in virgola mobile.

**ESERCIZIO 5 (5 punti)**

1. (2 punti) Spiegare in modo chiaro e sintetico le differenze tra memoria paginata e memoria segmentata.
2. (3 punti) Si consideri un sistema operativo con gestione della memoria segmentata e paginata. Ciascun job viene suddiviso al massimo in 3 segmenti. Ciascun segmento ha una dimensione massima di 128 pagine. Le pagine hanno dimensione 1KB. Si abbia ad esempio la seguente situazione:

Tabella dei Segmenti	
Segmento	Tabella delle Pagine
00	2
01	0
10	1

Tabella delle Pagine 0			
Pagina Virtuale	Pagina Fisica	Bit di Validità	Solo lettura
0000000	0001010	1	0
0000001	0010001	0	0
0000010	1011001	1	0
0000011	1011010	1	0
0000100	0011101	1	0
0000101	0101111	0	0
0000110	0110111	1	0
0000111	0100000	1	0
0001000	0100100	1	0
0001001	0001001	1	0

Tabella delle Pagine 1			
Pagina Virtuale	Pagina Fisica	Bit di Validità	Solo lettura
0000000	0000011	0	0
0000001	0010110	0	0
0000010	1001001	1	0
0000011	1001010	1	0
0000100	1010101	1	0
0000101	0011101	1	0
0000110	0111111	1	0
0000111	1011101	1	0
0001000	1010011	1	0
0001001	0001111	1	0
0001010	0011011	1	0
0001011	0100010	1	0

Tabella delle Pagine 2			
Pagina Virtuale	Pagina Fisica	Bit di Validità	Solo lettura
0000000	0100001	1	1
0000001	0101110	1	1
0000010	0110110	0	1
0000011	0000110	1	1
0000100	1100011	0	1
0000101	1000011	1	1
0000110	0010101	1	1

Il bit di validità indica se la pagina virtuale richiesta è presente (1) o no (0) nella memoria principale. Si considerino le seguenti richieste e mostrare: a quale indirizzo fisico corrispondono gli indirizzi virtuali, se l'operazione può essere conclusa con successo o se viene generato un errore (segmento non valido, pagina non valida, violazione di protezione) o un page fault:

- a. Read 0000000010101001101
  - b. Write 1000001000101011001
  - c. Read 0100001011001110011
  - d. Write 0000000111100100110
  - e. Read 0100010000001100001
  - f. Write 1110000100000100111
-

## ESERCIZIO 1

### Soluzione

Il circuito è formato da flip-flop di tipo JK, da porte logiche AND. Il circuito è sincronizzato da un segnale di clock che abilita le transizioni di stato nei flip-flop. Gli ingressi ai flip-flop sono infine controllati da un segnale 'count enable'. La tabella di eccitazione di un flip-flop JK è riportata in basso.

Q(t)	Q(t+1)	J	K
0	0	0	d
0	1	1	d
1	0	d	1
1	1	d	0

Ipotesizziamo che all'inizio tutti i flip-flop si trovino allo stato 0 e che il segnale di 'count enable' sia pari a 1. Dopo il primo colpo di clock, lo stato del primo flip flop passa da 0 a 1 ( $J = 1, K = 1$ ), dunque l'uscita A1 è pari a 1. Al secondo colpo di clock lo stato del primo flip flop torna a 0, mentre lo stato del secondo flip flop passa a 1 perché gli ingressi sono entrambi pari a 1 dovuti all'uscita precedente del primo flip flop (N.B. ovviamente la durata del clock dovrà essere tale da consentire la transizione di stato di un flip flop ma non la propagazione del nuovo stato ai flip flop successivi. Questi ultimi invece vedono in ingresso lo stato posseduto dai flip flop a cui sono collegati relativo all'istante precedente). Se proseguiamo nel ragionamento per i successivi 6 colpi di clock, osserveremo che le uscite A3 A2 A1 effettuano le seguenti transizioni:  $000 \Rightarrow 001 \Rightarrow 010 \Rightarrow 011 \Rightarrow 100 \Rightarrow 101 \Rightarrow 110 \Rightarrow 111$  (si osservi che il flip-flop A3 commuta solo quando lo stato precedente dei flip-flop A2 e A1 era pari a 1). Al nono colpo di clock tutte le variabili tornano a 0. La rete sequenziale dunque realizza un contatore binario sincronizzato dal clock.

## ESERCIZIO 2

### Soluzione

1. <Tag 8 bit> <Cache Index 5 bit> <Offset 3 bit>

2. e 3.

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  
1137 → 0001 0001 | 0011 0 | 111 → Block frame (551)<sub>10</sub> → Cache index 6  
AAAA → 1010 1010 | 1010 1 | 010 → Block frame (5461)<sub>10</sub> → Cache index 21

Se le parole venissero richieste sequenzialmente non si verificherebbe alcun hit.

### ESERCIZIO 3

#### Soluzione

$\$8 \leftarrow k*4$ ;  $\$9 \leftarrow \&v[0]+k*4$ ;  $\$4$ - $\$6$  vengono copiati in  $\$10$ - $\$12$

```
max_vettori: addi $29, $29, -24
              sw $8, 0($29)
              sw $9, 4($29)
              sw $10, 8($29)
              sw $11, 12($29)
              sw $12, 16($29)
              sw $31, 20($29)
              move $10, $4
              move $11, $5
              move $12, $6
              muli $8, $6, 4
              add $9, $8, $4
              addi $6, $6, 1
              lw $4, 0($9)
              beq $5, $6, loadV0 #se k+1 == N, carica v[0] in $5
                                #altrimenti carica v[k+1] in $5
cont:         jal media
              move $7, $6
              move $4, $10
              move $5, $11
              move $6, $12
              lw $8, 0($29)
              lw $9, 4($29)
              lw $10, 8($29)
              lw $11, 12($29)
              lw $12, 16($29)
              lw $31, 20($29)
              addi $29, $29, 24
              jr $31
loadV0:      lw $5, 0($10)
              j cont
```

## ESERCIZIO 4

### Soluzione.

1. Per rappresentare i numeri nel formato binario in virgola mobile è necessario innanzitutto rappresentare i due numeri nel formato binario in virgola fissa. Trattandosi di due numeri interi, si può procedere ad esempio con il metodo delle divisioni successive (cfr. Soluzioni del compito del 30/11/98). Il numero 193 si rappresenta in binario come 11000001, mentre il numero 104 è rappresentato come 1101000. La rappresentazione in virgola mobile nel formato IEEE 754 è la seguente:

193 (11000001  $\Rightarrow$  1.1000001 $\cdot$ 2<sup>7</sup>)

Segno	Esponente	Mantissa
Mantissa		
0	10000110	100000100000000000000000

104 (1101000  $\Rightarrow$  1.101000 $\cdot$ 2<sup>6</sup>)

Segno	Esponente	Mantissa
Mantissa		
0	10000101	101000000000000000000000

2. Algoritmo di somma:

**Confronto esponenti:** il numero 193 ha esponente maggiore, dunque il secondo numero deve essere denormalizzato in modo da eguagliare gli esponenti.

**Allineamento mantisse**

Per uguagliare gli esponenti, la mantissa del secondo numero va fatta scorrere di una posizione a destra (divisione per due). Pertanto le due mantisse (con esponente comune uguale a 7) diventano:

Primo numero: (1).1000001      Secondo numero: (0).1101

dove, fra parentesi, è stato evidenziato il valore del bit implicito

**Somma delle mantisse:**

(1).1000001 + (0).101 = (10).0101001

Questa mantissa non è normalizzata. Per normalizzarla nel formato IEEE 754 occorre far scorrere la mantissa di una posizione verso destra e aumentare l'esponente di una unità. Pertanto il risultato della somma verrà rappresentato nel calcolatore come (l'esponente è ora +8):

Segno	Esponente	Mantissa
Mantissa		
0	10000111	001010010000000000000000

Si può facilmente verificare che questo risultato corrisponde alla somma decimale 193+104=297

3. [cap.6, pag 39-40] La "floating point unit" (FPU) della ALU di un calcolatore può essere realizzata usando due unità aritmetiche in virgola fissa, una per l'esponente e una per la mantissa, che vengono accoppiate. L'unità per la mantissa si occupa di eseguire le operazioni aritmetiche di base sulla mantissa (somma algebrica, moltiplicazione e divisione). L'unità per l'esponente è più semplice in quanto deve eseguire solo operazioni di somma algebrica e di confronto fra numeri interi. Il confronto può essere effettuato attraverso una sottrazione degli esponenti.

Per eseguire la somma di due numeri in FP si fa la differenza degli esponenti. Il segno del risultato indica quale dei due esponenti è il più piccolo mentre il modulo indica il numero di scorrimenti verso destra che devono essere eseguiti sulla mantissa del numero più piccolo. Il registro che contiene la mantissa deve dunque essere del tipo a scorrimento. Gli scorrimenti possono essere pilotati da un contatore caricato con la differenza fra gli esponenti: ad ogni scorrimento il contatore viene decrementato finché non viene raggiunto il valore zero.

## ESERCIZIO 5

### **Soluzione.**

1. Vedi dispense del corso.
2.
  - a. Indirizzo fisico valido 01011100101001101
  - b. Indirizzo fisico valido 10101010101011001
  - c. Indirizzo fisico non valido (pagina non valida)
  - d. Violazione di scrittura sull'indirizzo fisico 00001101100100110
  - e. Indirizzo fisico valido 01001000001100001
  - f. Segmento non valido