

**PROVA SCRITTA DEL MODULO DI
CALCOLATORI ELETTRONICI
CORSO DI LAUREA IN INGEGNERIA BIOMEDICA
CORSO DI LAUREA IN INGEGNERIA ELETTRICA, ELETTRONICA ED INFORMATICA
12 luglio 2017**

MOTIVARE IN MANIERA CHIARA LE SOLUZIONI PROPOSTE A CIASCUNO DEGLI ESERCIZI SVOLTI

Cognome e Nome: _____ - **Matricola:** _____ - **Crediti:** _____

ESERCIZIO 1 (12 punti)

Sono date una memoria primaria di 1 KB e una memoria cache di 32 B. Le parole sono di 8 bit, ciascuna indirizzabile. L'organizzazione della memoria prevede blocchi di N parole (N potenza di 2).

- 1) (4 punti) Spiegare, precisando il significato e la funzione dei diversi campi, come vengono interpretati gli indirizzi logici per recuperare l'informazione contenuta nella cache nel caso in cui venga usata la modalità di indirizzamento diretto in funzione di N.
- 2) (4 punti) Calcolare l'hit ratio di cache in funzione di N ipotizzando la cache inizialmente vuota e chiamate alla sequenza di parole di indirizzo da 0 a 31 e da 80 a 95 per cinque volte consecutive.
- 3) (4 punti) Progettare la rete logica di confronto TAG, specificando e motivando se si tratta di una rete combinatoria o sequenziale. Si disegni il circuito nel dettaglio.

ESERCIZIO 2 (8 punti)

Si scriva il codice Assembler MIPS di una funzione `ordina` che, ricevendo in ingresso l'indirizzo iniziale di un vettore di interi `v` in `$4` e la sua dimensione `N` in `$5`, ordini gli elementi del vettore in ordine crescente. Nell'implementare questa funzione si supponga di utilizzare la funzione `minimo_in_testa` che, ricevendo l'indirizzo iniziale di un vettore su `$4` e la sua dimensione in `$5`, permuta il primo elemento del vettore con quello minore.

Il codice MIPS potrebbe implementare il seguente codice C:

```
void ordina(int* v, int N)
{
    int i, n;

    for (i=0, n=N; i<N-1; i++, n--)
        minimo_in_testa(&v[i], n);
}
```

ESERCIZIO 3 (7 punti)

Sia data la seguente stringa di 8 bit: 01011100 (il bit più significativo è a destra).

- 1) (3 punti) Codificare la stringa con il codice di Hamming.
- 2) (4 punti) Ipotizzare un errore sul bit meno significativo della stringa data ed eseguire i passi per la correzione dello stesso.

ESERCIZIO 4 (6 punti)

L'esecuzione di un'istruzione al calcolatore richiede 5 cicli di clock, dei quali 2 non tengono impegnato il bus di sistema. Durante l'esecuzione di alcuni test viene rilevato che una frazione p della totalità dell'istruzione rate, ovviamente compresa tra 0 ed 1, è utilizzata per istruzioni che non richiedono trasferimenti di I/O.

Volendo utilizzare un modulo DMA in modalità "transparent" per i trasferimenti di I/O, indicare in funzione di p il numero medio di trasferimenti per istruzione, nell'ipotesi che lungo il bus dati possa essere trasferita una parola per ciclo di clock con il DMA. Mettere in grafico tale numero in funzione di p .

ESERCIZIO 1

Essendo la memoria di 1 KB, il numero di bit di indirizzamento complessivi è pari a 10 in quanto ciascuna parola è pari a 8 bit, ovvero 1 B. Poiché la cache è di $32 \text{ B} = 2^5 \text{ B}$ e ciascun blocco/linea è formata da N parole, la suddivisione in TAG, cache index e offset è data da:

TAG: 5 bit

Cache index: $5 - \log_2 N$

Offset: $\log_2 N$

E' evidente che N è nell'intervallo compreso tra 1 e 32, ovvero entro i possibili valori 1, 2, 4, 8, 16, 32 che portano rispettivamente ai valori di offset pari a 0, 1, 2, 3, 4, 5.

Per il calcolo dell'hit ratio, ragioniamo come segue. Innanzi tutto, l'intervallo di chiamata è dato da 32 parole nel primo gruppo di chiamate (0-31) e 16 nel secondo (80-95). Ora, la nostra attenzione va a porsi sulle configurazioni degli indirizzi: l'indirizzo della prima parola del primo gruppo di chiamate si scrive come 0000000000 in binario e dell'ultima come 0000011111; l'indirizzo della prima parola del secondo gruppo si scrive come 0001010000 e dell'ultima come 0001011111. Eliminando i TAG da tutte si ha che il C.I. in dipendenza di N è al massimo 00000 per la prima parola e 11111 per l'ultima del primo gruppo, 10000 per la prima parola e 11111 per l'ultima del secondo gruppo. Ciò significa che, indipendentemente dalla grandezza dei blocchi/linee, viene copiato un certo numero di blocchi di primaria con medesimo C.I. nelle stesse linee di cache. La cache viene totalmente riempita dal primo gruppo (la cache contiene al massimo 32 parole), mentre il secondo gruppo verrà mappato a partire dalla seconda metà della memoria cache, com'è evidente dal C.I. della prima parola del secondo gruppo.

Sulla base di queste considerazioni:

- nel caso di $N=32$, abbiamo un'unica linea di cache disponibile costituita da 32 parole. Viene copiato quindi tutto il primo gruppo con un numero di hit pari 31, poi viene sovrascritto dal secondo gruppo (incluse le 16 parole precedenti), con un hit pari 15. Questi stessi valori si ripetono per altre quattro volte per cui $H_c = 46/48 = 0,96$.
- nei casi intermedi con $N=2,4,8,16$, abbiamo $32/N$ blocchi chiamati nel primo gruppo e $16/N$ blocchi chiamati nel secondo gruppo, per ciascuno dei quali si hanno $N-1$ hit alla prima chiamata. Dalla seconda chiamata in poi, per la prima metà del primo gruppo, quindi su $16/N$ blocchi, si hanno N hit, mentre per i restanti si hanno ancora $N-1$ hit. Abbiamo dunque, su $48 \cdot 5$ chiamate totali, $H_c = [(48/N) \cdot (N-1) + 4 \cdot 16 + 4 \cdot (32/N) \cdot (N-1)] / (48 \cdot 5)$.
- nel caso di $N=1$, abbiamo tante linee quante sono le parole, per cui avremo 0 hit per la prima chiamata ed entrambi i gruppi, dalla seconda chiamata in poi avremo 16 hit soltanto. Quindi $H_c = (16 \cdot 4) / (5 \cdot 48) = 0,27$. Si noti che la formula del punto precedente è valida anche in questo caso.

Il circuito per il confronto dei TAG è una rete logica combinatoria formato da cinque porte XOR per il confronto di ciascuno dei due gruppi di bit di TAG (linea memorizzata vs. blocco chiamato) al primo livello e una porta OR al secondo livello che riceve come input le quattro uscite delle porte XOR. Poiché la XOR restituisce 1 quando i due bit sono diversi, basta che almeno una delle cinque porte XOR restituisca 1 per evidenziare che la parola non si trova in cache, ovvero l'OR restituisce 1. Volendo si può negare l'uscita della OR, utilizzando quindi una porta NOR, per avere 1 nel caso contrario, ovvero se i due TAG sono identici.

Si lascia allo studente il disegno del circuito.

ESERCIZIO 2

```
$8 ← i;
$13 ← copia di &v[0]; $14 ← N-1

ordina:      addi $29, $29, -16      #salvataggio contesto
             sw $8, 0($29)
             sw $13, 4($29)
             sw $14, 8($29)
             sw $31, 12($29)
             move $13, $4
             subi $14, $5, 1
             move $8, $0            #i=0
for:         beq $8, $14, exit
             jal minimo_in_testa
             addi $8, $8, 1
             addi $4, $4, 4
             subi $5, $5, 1
             j for
exit:        move $4, $13
             addi $5, $13, 1
             lw $8, 0($29)          #ripristino contesto
             lw $13, 4($29)
             lw $14, 8($29)
             lw $31, 12($29)
             addi $29, $29, 16
             jr $31                #ritorno a chiamante
```

ESERCIZIO 3

La stringa da codificare è formata da 8 bit, che indichiamo con N. Poiché deve sussistere:

$$2^K - 1 \geq N$$

Si ha che K che soddisfa la relazione è pari a 4, ovvero sono necessari quattro bit di controllo che indicheremo con c_0, c_1, c_2, c_3 .

Per completare il calcolo si applicano le formule della codifica di Hamming ovvero:

$$c_0 = \text{XOR}(b_0, b_1, b_3, b_4, b_6) = 1$$

$$c_1 = \text{XOR}(b_0, b_2, b_3, b_5, b_6) = 0$$

$$c_2 = \text{XOR}(b_1, b_2, b_3, b_7) = 0$$

$$c_3 = \text{XOR}(b_4, b_5, b_6, b_7) = 0$$

Poiché b_0 è controllato da c_0 e c_1 , è evidente che vengono alterati solo quei due bit, il ricalcolo dei quali condurrebbe alla stringa di errore 1100, corrispondente appunto al bit b_0 .

ESERCIZIO 4

Poiché per ogni trasferimento è richiesto un ciclo di clock, per la frazione p possono essere trasferite solo 2 parole per istruzione, mentre per la restante possono essere trasferite ben 5 parole.

Il numero medio di trasferimenti $T(p)$ è dato dalla formula:

$$T(p) = 2 \cdot p + 5 \cdot (1 - p) = 5 - 3 \cdot p$$

Si tratta di una retta con p compreso tra 0 ed 1:

