

**PROVA SCRITTA DEL MODULO DI**  
**CALCOLATORI ELETTRONICI**

Corsi di Laurea in:  
INGEGNERIA ELETTRICA, ELETTRONICA ED INFORMATICA  
INGEGNERIA BIOMEDICA  
11 settembre 2019

**NOME:**

**COGNOME:**

**MATRICOLA:**

**CFU:**

**ESERCIZIO 1 (7 punti)**

- 1) (3 punti) Progettare una rete logica che, date due stringhe binarie  $A$  e  $B$  di  $n$  bit, presenti un'uscita  $Z$  pari ad 1 se  $A > B$ . Disegnare il circuito.
- 2) (4 punti) Progettare una ALU che a partire da tre bit di controllo  $s_2, s_1, s_0$  e un parallel adder realizzi attraverso un'opportuna rete logica le funzioni di  $A$  e  $B$  indicate dalla tabella di verità a lato. Disegnare il circuito ottenuto.

$s_2$	$s_1$	$s_0$	F
0	0	0	-A
0	0	1	-B
0	1	0	A+B
0	1	1	A-B
1	0	0	B-A
1	0	1	A-1
1	1	0	B-1
1	1	1	Don't care

**ESERCIZIO 2 (8 punti)**

Si consideri una memoria primaria costituita da 1 KB e una memoria cache di 128 B, con blocchi di 8 B. E' possibile indirizzare il singolo byte.

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 nel caso di indirizzamento associativo su insiemi a quattro vie.
2. (3 punti) Supponendo la cache inizialmente vuota, si considerino le chiamate ai seguenti indirizzi (espressi in decimale): da 0 a 55, da 80 a 95 in questo ordine, per dieci volte consecutive. Indicare lo stato finale della cache.
3. (2 punti) Si calcoli l'hit ratio di cache sulla base delle chiamate al punto precedente.
4. (1 punto) Si consideri infine una gerarchia di memoria a due livelli costituita da cache e primaria. Il tempo di accesso in cache è pari a 2 nsec mentre quello in primaria è pari a 40 nsec. Calcolare il tempo medio di accesso alla gerarchia.

**ESERCIZIO 3 (14 punti)**

Implementare una procedura Assembly MIPS che, dati l'indirizzo iniziale di un vettore  $v$  in \$4, la sua dimensione  $N$  in \$5, e un valore numerico  $t$  in \$6, restituisca in \$7 il più lontano valore a  $t$  contenuto in  $v$ . Per implementare la funzione si faccia uso della funzione `abs` che, ricevendo un valore numerico in \$4, restituisce il suo valore assoluto in \$5.

**ESERCIZIO 4 (4 punti)**

Descrivere in modo chiaro e sintetico la differenza tra gestione dell'I/O attraverso interruzioni e attraverso DMA.

## ESERCIZIO 1

- 1) La rete richiesta presenta due livelli di logica: il primo è costituito da  $n$  porte XOR che ricevono in ingresso le coppie  $A_i$  e  $B_i$  delle due stringhe ( $i=0, \dots, n-1$ ). Ciascuna di queste porte permette di sapere se i bit differiscono. Il secondo livello è una semplice porta OR a  $n$  ingressi. La OR produrrà uscita 1 se almeno una delle porte XOR precedenti avrà dato valore 1. Si lascia allo studente il disegno del circuito.
- 2) Per implementare l'ALU richiesta, è necessario anteporre agli ingressi  $i$ -esimi del parallel adder relativi al primo e secondo operando una rete combinatoria che forzi i bit di  $A$  e  $B$  a implementare la funzione richiesta in corrispondenza dei bit di controllo. Questi bit governano anche il valore del riporto in ingresso al parallel adder,  $cin$ .

s2	s1	s0	F	Ai	Bi	Cin
0	0	0	-A	A'	0	1
0	0	1	-B	0	B'	1
0	1	0	A+B	A	B	0
0	1	1	A-B	A	B'	1
1	0	0	B-A	A'	B	1
1	0	1	A-1	A	1	0
1	1	0	B-1	1	B	0
1	1	1	Don't care	D	D	D

Ottenuta la tabella di verità per le tre reti, è sufficiente semplificarle con le mappe di Karnaugh. Mostriamo qui la sola rete relativa ai bit di  $A$ , lasciando le altre due e il disegno del circuito allo studente.

$$A_i^{new} = S_2 S_1 + S_1 A_i + S_1' S_0' A_i' + S_2 S_0 A_i$$

	00	01	11	10
00	1		1	1
01		1	1	
11		1	D	1
10			D	

## ESERCIZIO 2

1. I tre bit meno significativi compongono dell'indirizzo di primaria compongono l'offset (i blocchi sono di otto parole). Il numero di linee di cache è invece 16 (128 B/8 B), per cui il cache index è formato da 4 bit. Poiché il metodo di indirizzamento è associativo su insiemi a quattro vie, il cache index cede i due bit più significativi al TAG ottenendo la sequenza TAG-Set Index-Offset pari a: 5-2-3.
2. Dalla lettura del testo si nota che le prime 56 chiamate appartengono a parole contigue a partire dal blocco 0 di primaria, per un totale di 7 blocchi. Per definizione di metodo set-associativo, ciascuno di questi blocchi presenta in set index diverso (da 0 a 3) e vengono così allocati nelle prime linee libere di ciascun insieme di cache.  
Il secondo gruppo di chiamate, da 80 a 95, è costituito da 16 parole contigue. Dividendo l'indirizzo della prima per il numero di blocchi si ottiene dal resto pari a 0 che essa è la prima parola di un blocco, e l'ulteriore divisione del block frame (10) per il numero di insiemi ci fa calcolare a quale insieme va allocato il blocco con le prime parole, permettendoci anche di sapere l'indirizzo di set del blocco successivo. Il valore ottenuto è 2, quindi il blocco 80-87 viene allocato nella terza linea libera del set 2 e il blocco 88-95 nella seconda linea libera del set 3.  
Le successive chiamate ripetono la sequenza dall'inizio e non determina variazioni di allocazione perché tutte le parole sono già in cache. Schematizzando la cache come segue, lo stato finale è dunque:

Index di set	Linea per set			
	0	1	2	3
0	0-7	32-39		
1	8-15	40-47		
2	16-23	48-55	80-87	
3	24-31	88-95		

3. Abbiamo 7 hit per ciascuno degli 8 blocchi chiamati al primo ciclo, mentre dal secondo ciclo abbiamo tutti hit. Si ottiene dunque:

$$H_C = \frac{7 \cdot 9 + 9 \cdot 8 \cdot 9}{10 \cdot 8 \cdot 9} = \frac{63 + 648}{720} = 0.9875$$

4. La formula del tempo medio di accesso alla gerarchia di memoria data è:

$$\hat{T} = T_C + (1 - H_C) \cdot T_P$$

Sostituendo i valori di  $H_C$  trovati nel precedente esercizio si ha:

$$\hat{T}_{SetAssociativo} = 2 + 0.0125 \cdot 40 = 2 + 0.5 = 2.5nsec$$

### ESERCIZIO 3

#### Soluzione

$\$4 \leftarrow \&v[0]; \$5 \leftarrow N; \$6 \leftarrow t$

```
piuvicino:  addi $29, $29, -28    #salvataggio del contesto
            sw $8, 0($29)
            sw $9, 4($29)
            sw $10, 8($29)
            sw $11, 12($29)
            sw $12, 16($29)
            sw $13, 20($29)
            sw $31, 24($29)
            move $10, $4          #copia di &v[0] e N
            move $11, $5
            lw $7, 0($10)         #il valore più vicino è v[0]
            move $4, $7          #passaggio dei parametri per abs
            sub $4, $4, $6        #parametro in $4: v[0]-t
            jal abs               #$5 ← |v[0]-t|
            move $12, $5         #lo copio in $12
            addi $8, $0, 1        #i=1
for:        beq $8, $11, exit     #for i<N
            muli $9, $8, 4
            addi $9, $9, $10      #$9 ← &v[i]
            lw $4, 0($9)         #$4 ← v[i]
            move $13, $4         #lo copio in $13
            sub $4, $4, $6        #parametro in $4: v[i]-t
            jal abs               #$5 ← |v[i]-t|
            slt $9, $12, $5       #$9 ← $12 < $5 ?
            beq $9, $0, upd       #se no, prossima iterazione
            move $12, $5         #altrimenti, aggiorno il più lontano
            move $7, $13         #il più lontano in $7
upd:        addi $8, $8, 1
            j for
exit:       move $4, $10         #ripristino dei parametri in ingresso
            move $5, $11
            sw $8, 0($29)
            sw $9, 4($29)
            sw $10, 8($29)
            sw $11, 12($29)
            sw $12, 16($29)
            sw $13, 20($29)
            sw $31, 24($29)
            addi $29, $29, 28
            jr $31              #ritorno al chiamante
```

### ESERCIZIO 4

Vedi dispense del corso o libro di testo.