

**PROVA SCRITTA DEL MODULO DI
CALCOLATORI ELETTRONICI
NUOVO E VECCHIO ORDINAMENTO DIDATTICO**
5 Febbraio 2015

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

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

1.1 Progettare nei seguenti due modi una ALU che esegua le operazioni in tabella su due operandi A e B ad n bit:

s1	s0	c = 0	c = 1
0	0	A+B	A+B+1
0	1	A	A+1
1	0	B	B+1
1	1	A-B+1	A-B

dove **s1** ed **s0** sono due variabili di controllo.

a) (NO: 4 punti - VO: 2 punti) Si utilizzi un parallel adder ad n bit e una opportuna rete logica. Si tenga conto che **c** è il bit di riporto del parallel adder.

b) (NO: 4 punti - VO 2 punti) Si sostituisca la rete logica al passo precedente con due MUX 4-1.

1.2 (solo VO: 3 punti) Disegnare lo schema logico di un parallel adder in termini dei full adder che lo costituiscono e calcolarne il tempo di ritardo, assumendo che quello di un singolo full adder sia pari a **d**.

ESERCIZIO 2 (NO: 10 punti - VO: 8 punti)

1. (NO: 3 punti – VO: 1 punto) Si consideri una memoria primaria costituita da 1024 parole e una memoria cache costituita da 256 parole. Si mostri il formato degli indirizzi nei casi in cui il metodo di indirizzamento sia rispettivamente:

- a. associativo su insiemi a 2 vie con blocchi di 2 parole;
- b. diretto con blocchi di 16 parole;
- c. diretto con blocchi di 64 parole.

2. (NO: 7 punti – VO: 5 punti) Si indichino lo stato finale della cache e l'hit ratio per ognuno dei metodi precedenti, sapendo che il processore effettua le chiamate da 0 a 75 (compresi) per due volte (0, 1, ... 75, 0, 1, ... 74, 75).

3. (solo VO: 2 punti) Si indichino brevemente le motivazioni che giustificano l'uso di una gerarchia di memoria.

ESERCIZIO 3 (8 punti)

Sia dato il seguente codice Assembler.

- a) (4 punti) Indicare eventuali errori di sintassi/semantica presenti nel codice.
- b) (4 punti) Analizzare il codice spiegando chiaramente qual è la funzione svolta dallo stesso, quali sono i parametri in ingresso e quali in uscita.

```
funzione: subi $29, $29, -4
           sw $8, $29(0)
           addi $6, $0, 1
           move $8, $0
a:         beq $8, $5, b
           muli $6, $6, $4
           add $8, $8, 1
           jr a
b:         sw $8, $29(0)
           addi $29, $29, 4
           j $31
```

ESERCIZIO 4 (NO: 7 punti – VO: 6 punti)

Le parole trasferite a/dalla memoria di un calcolatore sono codificate utilizzando il codice di Hamming. Si consideri la stringa di 13 bit 1010011011101 (il bit meno significativo è a sinistra), risultato della codifica di una parola di N bit secondo il codice di Hamming.

1. (VO: 1 punto – NO: 2 punti) calcolare N, supponendo di aver fatto uso del numero minimo di bit di controllo necessari.
2. (2 punti) scrivere la parola di N bit a partire dalla stringa data;
3. (3 punti) indicare eventuali errori nella stringa codificata, specificando quale dei bit è stato alterato.

ESERCIZIO 5 (solo VO: 4 punti)

Descrivere il concetto di pipeline nelle architetture parallele.

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

1.1 Progettare nei seguenti due modi una ALU che esegua le operazioni in tabella su due operandi A e B ad n bit:

$s1$	$s0$	$c = 0$	$c = 1$
0	0	A+B	A+B+1
0	1	A	A+1
1	0	B	B+1
1	1	A-B+1	A-B

dove $s1$ ed $s0$ sono due variabili di controllo.

c) (NO: 4 punti - VO: 2 punti) Si utilizzi un parallel adder ad n bit e una opportuna rete logica. Si tenga conto che c è il bit di riporto del parallel adder.

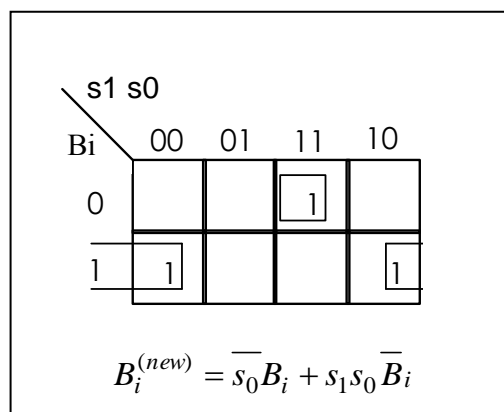
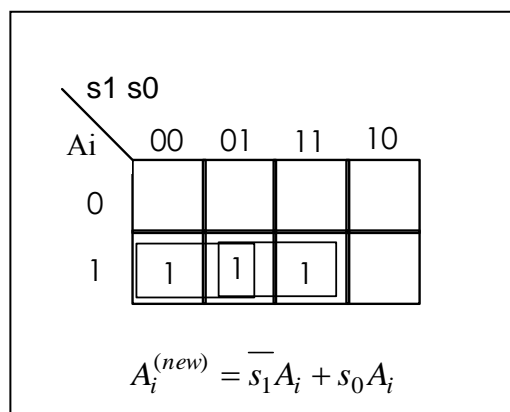
d) (NO: 4 punti - VO: 2 punti) Si sostituisca la rete logica al passo precedente con due MUX 4-1.

1.2 (solo VO: 3 punti) Disegnare lo schema logico di un parallel adder in termini dei full adder che lo costituiscono e calcolarne il tempo di ritardo, assumendo che quello di un singolo full adder sia pari a d .

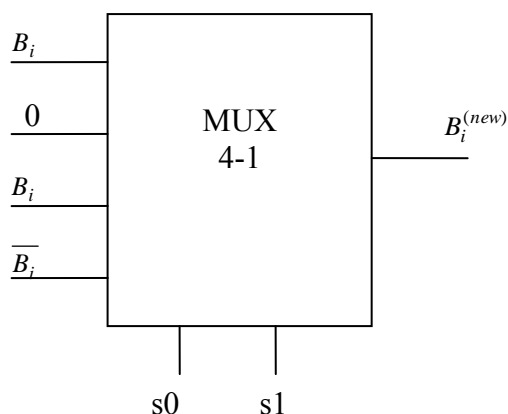
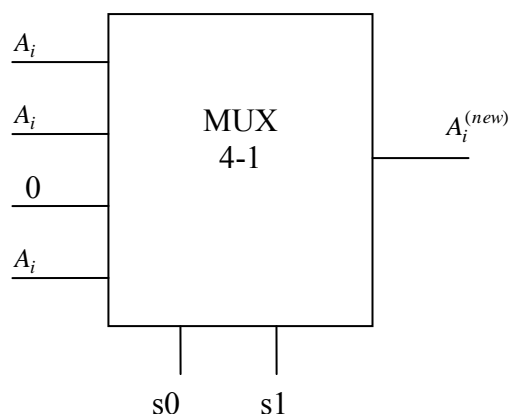
Soluzione

Punto a).

Una semplice ispezione visiva della tabella ci conduce immediatamente alle mappe di Karnaugh relative alle reti logiche a monte del parallel adder. Indicando con A_i e B_i l' i -esimo bit del primo e del secondo addendo, si calcolano le forme minime per l'espressioni degli addendi in ingresso al parallel adder.



Punto b). In questo caso è sufficiente connettere gli addendi nella forma voluta (diretta o negata, oppure in forma costante), ai quattro ingressi di ciascun multiplexer, lasciando il compito di selezionare quelli opportuni alle due variabili di controllo.



ESERCIZIO 2 (NO: 10 punti - VO: 8 punti)

4. (NO: 3 punti – VO: 1 punti) Si consideri una memoria primaria costituita da 1024 parole e una memoria cache costituita da 256 parole. Si mostri il formato degli indirizzi nei casi in cui il metodo di indirizzamento sia rispettivamente:
- associativo su insiemi a 2 vie con blocchi di 2 parole;
 - diretto con blocchi di 16 parole;
 - diretto con blocchi di 64 parole.
5. (NO: 7 punti – VO: 5 punti) Si indichino lo stato finale della cache e l'hit ratio per ognuno dei metodi precedenti, sapendo che il processore effettua le chiamate da 0 a 75 (compresi) per due volte (0, 1, ... 75, 0, 1, ... 74, 75).
6. (solo VO: 2 punti) Si indichino brevemente le motivazioni che giustificano l'uso di una gerarchia di memoria.

Soluzione

1. Il campo indirizzi avrà ampiezza 10 bit (devo indirizzare 1024 parole).
- Avremo 2 insiemi da 64 blocchi ciascuno (la cache contiene 256 parole, quindi 128 blocchi, visto che ogni blocco contiene 2 parole):
< TAG 8 bit > < Set Index 1 bit > < Offset 1 bit >
 - Avremo 256 parole / (16 parole/blocco) = 16 blocchi, quindi:
< TAG 2 bit > < Cache Index 4 bit > < Offset 4 bit >
 - Avremo 256 parole / (64 parole/blocco) = 4 blocchi, quindi:
< TAG 2 bit > < Cache Index 2 bit > < Offset 6 bit >
- 2.
-

	0	1	2	3	...	74	75	0	1	...	75
BF	0	0	1	1		37	37	0	0		37
S.I.	0	0	1	1		1	1	0	0		1
Hit		X		X			X	X	X	X	X

Si noti come per la metà delle prime 76 chiamate (0...75) si abbia un hit.

Durante le 76 chiamate successive si avrà sempre hit, perché tutte le pagine sono già in cache (non c'è mai rimpiazzamento).

$$\text{Hit Ratio} = (76/2 + 76)/(2*76) = 3/4 = 0.75$$

Stato finale della cache:

Set 0						Set 1					
Blocco 0	Blocco 1	...	18	...	63	0	1	...	18	...	63
Ind. 0, 1	Ind. 4, 5		72, 73			2, 3	6, 7		74, 75		

Occuperò 36 blocchi in tutto (perché sto chiamando 76 parole diverse), equamente ripartiti in 18 per insieme. I restanti saranno vuoti.

-

	0	1	...	15	16	...	31	...	63	64	...	74	75	0	...	75
BF	0	0	0	0	1	1	1	...	3	4	4	4	4	0		4
C.I.	0	0	...	0	1	...	1		3	4	...	4	4	0		4
Hit		X	X	X		X	X		X		X	X	X	X	X	X

Per contare gli hit dobbiamo analizzare quello che succede nella due metà delle chiamate. Nella prima metà, dopo il primo miss si verificano 15 hit (una volta che il blocco è portato in cache, le 15 parole successive alla prima danno un hit), e questo succede 4 volte (fino a 63). Nell'ultimo caso si verifica un cache miss su 64 e successivamente 11 hit (da 65 a 75 inclusi).

Durante le 76 chiamate successive si avrà sempre hit, perché tutte le pagine sono già in cache (non c'è mai rimpiazzamento).

$$\text{Hit Ratio} = (15 \cdot 4 + 11 + 76) / (2 \cdot 76) = 147 / 152 = 0.97$$

Stato finale della cache:

Blocco 0	1	2	3	4	...
0...15	16...31	32...47	48...63	64...75	

Occuperò completamente 4 blocchi (0...3) e parte del blocco 4.

C.

	0	1	...	63	64	65		75	0	...	75
BF	0	0	0	0	1	1	1	1	0		4
C.I.	0	0	...	0	1	1	1	1	0		4
Hit		X	X	X		X	X	X	X	X	X

Per contare gli hit dobbiamo analizzare quello che succede nella due metà delle chiamate. Nella prima metà, dopo il primo miss si verificano 63 hit (una volta che il blocco è portato in cache, le 63 chiamate successive alla prima danno un hit), e questo succede 1 volte (fino a 63). Nell'ultimo caso si verifica un cache miss su 64 e successivamente 11 hit (da 65 a 75 inclusi).

Durante le 76 chiamate successive si avrà sempre hit, perché tutte le pagine sono già in cache (non c'è mai rimpiazzamento).

$$\text{Hit Ratio} = (63 + 11 + 76) / (2 \cdot 76) = 150 / 152 = 0.99$$

Stato finale della cache:

Blocco 0	1	...
0...63	64...75	

Occuperò completamente il blocco 0 e parte del blocco 1.

3. Vedi dispense del corso.

ESERCIZIO 3 (solo NO: 8 punti)

Sia dato il seguente codice Assembler.

c) (4 punti) Indicare eventuali errori di sintassi/semantica presenti nel codice.

d) (4 punti) Analizzare il codice spiegando chiaramente qual è la funzione svolta dallo stesso, quali sono i parametri in ingresso e quali in uscita.

```
funzione:  subi $29, $29, -4
           sw $8, $29(0)
           addi $6, $0, 1
           move $8, $0
a:         beq $8, $5, b
           muli $6, $6, $4
           add $8, $8, 1
           jr a
b:         sw $8, $29(0)
           addi $29, $29, 4
           j $31
```

Soluzione.

Nel codice sono presenti evidenti errori di sintassi, laddove ad esempio nell'istruzione `sw` vengono permutate la posizione del secondo e terzo operando, e di semantica, laddove un'istruzione con indirizzamento immediato (`addi`) viene usata con indirizzamento a registro (`add`), e viceversa. Un altro errore consiste nell'uso errato dell'istruzione `j`, che prevede come operando un indirizzo di memoria, e dell'istruzione `jr`, che invece richiede un registro. Un ultimo errore è presente nella sezione di ripristino del contesto: viene usata l'istruzione di memorizzazione `sw` invece di quella di prelievo `lw`.

Funzione data con errori (in neretto)	Funzione corretta
<pre>funzione: subi \$29, \$29, -4 sw \$8, \$29(0) addi \$6, \$0, 1 move \$8, \$0 a: beq \$8, \$5, b muli \$6, \$6, \$4 add \$8, \$8, 1 jr a b: sw \$8, \$29(0) addi \$29, \$29, 4 j \$31</pre>	<pre>funzione: addi \$29, \$29, -4 sw \$8, 0(\$29) addi \$6, \$0, 1 move \$8, \$0 a: beq \$8, \$5, b mul \$6, \$6, \$4 addi \$8, \$8, 1 j a b: lw \$8, 0(\$29) addi \$29, \$29, 4 jr \$31</pre>

Una volta che la funzione è stata emendata dagli errori, si può notare che essa, dopo aver inizializzato `$6` ad 1, lo aggiorna moltiplicandolo per il contenuto di `$4`. Ciò viene fatto finché il valore presente in `$8` è minore di quello presente in `$5`.

La presente funzione realizza quindi l'elevamento a potenza, con base in `$4` ed esponente in `$5`. Il risultato viene memorizzato in `$6`.

ESERCIZIO 4 (NO: 7 punti – VO: 6 punti)

Le parole trasferite a/dalla memoria di un calcolatore sono codificate utilizzando il codice di Hamming. Si consideri la stringa di 13 bit 1010011011101 (il bit meno significativo è a sinistra), risultato della codifica di una parola di N bit secondo il codice di Hamming.

4. (VO: 1 punto – NO: 2 punti) calcolare N, supponendo di aver fatto uso del numero minimo di bit di controllo necessari.
5. (2 punti) scrivere la parola di N bit a partire dalla stringa data;
6. (3 punti) indicare eventuali errori nella stringa codificata, specificando quale dei bit è stato alterato.

Soluzione.

- 1) Deve essere rispettata la condizione:

$$2^K \geq N + K + 1 \quad (1),$$

dove K è il numero di bit di controllo inseriti. Essendo $N + K = 13$, si evince dalla (1) che il numero minimo di bit di controllo richiesto è 4. Da cui $N = 8$.

- 2) Nella codifica di Hamming, la sequenza in ingresso presenta la seguente struttura:

c_0	c_1	b_0	c_2	b_1	b_2	b_3	c_3	b_4	b_5	b_6	b_7	b_8
1	0	1	0	0	1	1	0	1	1	1	0	1

Dove $c_0 \dots c_3$ sono i 4 bit costituenti il vettore di controllo, e $b_0 \dots b_8$ i 9 bit trasmessi. La sequenza ricevuta è 101111101.

- 3) Per verificare la presenza di un errore, dobbiamo ricalcolare il vettore di controllo a partire dalla sequenza ricevuta. Si ha:

$$c'_0 = b_0 \oplus b_1 \oplus b_3 \oplus b_4 \oplus b_6 \oplus b_8 = 1$$

$$c'_1 = b_0 \oplus b_2 \oplus b_3 \oplus b_5 \oplus b_6 = 1$$

$$c'_2 = b_1 \oplus b_2 \oplus b_3 \oplus b_7 \oplus b_8 = 1$$

$$c'_3 = b_4 \oplus b_5 \oplus b_6 \oplus b_7 \oplus b_8 = 0$$

Il passo successivo è calcolare il vettore di errore dato dalla differenza dei vettori di controllo c e c' (ricordiamo che somma e differenza tra bit producono lo stesso risultato):

$$e_0 = c_0 \oplus c'_0 = 0$$

$$e_1 = c_1 \oplus c'_1 = 1$$

$$e_2 = c_2 \oplus c'_2 = 1$$

$$e_3 = c_3 \oplus c'_3 = 0$$

Poiché il vettore risultante 0110 non è nullo, vi è un errore nella stringa di 13 bit data e precisamente nella posizione indicata dal vettore di errore tradotto in notazione decimale. Il bit sbagliato è quindi il sesto bit (b_2), e la parola corretta è 100111101.

ESERCIZIO 5 (solo VO: 4 punti)

Descrivere il concetto di pipeline nelle architetture parallele.

Soluzione

Vedi dispense del corso su "Architetture Parallele".