

PROVA SCRITTA DEL CORSO DI
CALCOLATORI ELETTRONICI
NUOVO ORDINAMENTO DIDATTICO
19 Giugno 2008

NOME:

COGNOME:

MATRICOLA:

ESERCIZIO 1 (NO: 7 punti – VO: 6 punti)

Progettare una ALU che esegua le seguenti operazioni su due operandi A e B ad n bit:

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

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

- (NO: 5 punti- VO: 4 punti) Si utilizzi un parallel adder ad n bit e le opportune reti logiche. Si tenga conto che **c** è il bit di riporto del parallel adder.
- (2 punti) Si sostituiscano la reti logiche al passo precedente con dei MUX 4-1.

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

Si consideri un calcolatore che dispone di una memoria principale di 256 Mbyte e di una memoria cache di 512 Kbyte. E' possibile accedere al singolo byte e la memoria è suddivisa in blocchi da 16 byte.

- (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 venga usata la modalità di indirizzamento:
 - diretto;
 - associativo su insiemi, in cui ciascun insieme contenga due blocchi.
- (VO: 6 punti - NO: 8 punti) Si consideri la cache di cui alla domanda precedente. Ipotizzare che il processore acceda in sequenza ai byte dall'indirizzo 0000000 a 00007FF e da 0180000 a 01807FF in questo ordine, e ripeta questa sequenza di accesso per 5 volte consecutive. Si ipotizzi inoltre che la cache sia inizialmente vuota. Calcolare il numero di miss sia nel caso di modalità ad indirizzamento diretto sia nel caso di modalità ad indirizzamento associativo su insiemi spiegata nel punto precedente. Calcolare quindi l'hit ratio per i due casi.

ESERCIZIO 3 (solo NO: 9 punti)

Scrivere una funzione Assembly MIPS, di nome `copy_idx`, che, dati in ingresso due vettori di dimensione N , `idx` e `x`, riordini gli elementi di `x` in un nuovo vettore `y`, secondo gli indici passati in `idx`. Più precisamente, il generico elemento `y[idx[i]]` dovrà contenere l'elemento `x[i]`.

Esempio per $N=8$: `idx`: 1, 2, 3, 4, 0, 5, 7, 6

`x`: 21, 32, 45, 51, 10, 67, 88, 73.

 Il risultato `y` conterrà: 10, 21, 32, 45, 51, 67, 73, 88.

Si supponga che `x` (`&x[0]`) sia passato alla funzione in `$4`, `idx` (`&idx[0]`) in `$5`, `y` (`&y[0]`) in `$6` e N in `$7`. Si abbia cura di ripristinare il contenuto di tutti i registri temporanei usati nella funzione e dei registri `$4`, `$5`, `$6`, `$7` (modalità callee save).

ESERCIZIO 3 (solo VO: 7 punti)

Si supponga di disporre di tre macchine: a pila, a uno e a due indirizzi. Per ognuna di queste si abbiano le seguenti istruzioni:

A pila		A un indirizzo		A due indirizzi	
Istruzione	Semantica	Istruzione	Semantica	Istruzione	Semantica
PUSH X	$M[X] \rightarrow \text{push}$	STORE X	$\text{ACC} \rightarrow M[X]$	MOV X1,X2	$M[X1] \rightarrow M[X2]$
POP X	$\text{pop} \rightarrow M[X]$	LOAD X	$M[X] \rightarrow \text{ACC}$	ADD X1,X2	$M[X1] + M[X2] \rightarrow M[X2]$
ADD	$\text{pop} + \text{pop} \rightarrow \text{push}$	ADD X	$\text{ACC} + M[X] \rightarrow \text{ACC}$	DIV X1,X2	$M[X1] / M[X2] \rightarrow M[X2]$
DIV	$\text{pop}(2) / \text{pop}(1) \rightarrow \text{push}$	DIV X	$\text{ACC} / M[X] \rightarrow \text{ACC}$	SUB X1,X2	$M[X1] - M[X2] \rightarrow M[X2]$
SUB	$\text{pop}(2) - \text{pop}(1) \rightarrow \text{push}$	SUB X	$\text{ACC} - M[X] \rightarrow \text{ACC}$		

ACC è il registro accumulatore; $M[X]$ indica il dato nella locazione di memoria X.

- (4 punti) Facendo attenzione a non sovrascrivere i contenuti iniziali della memoria, si scriva, per ognuna delle tre macchine, la sequenza delle istruzioni necessarie per realizzare la seguente operazione:

$$Z = A / (B - C) + D$$

(suggerimento: si usi uno o più registri dove depositare i risultati parziali)

- (3 punti) Spiegare in modo chiaro e sintetico i vari tipi di indirizzamento di un'istruzione.

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

I trasferimenti di parole a/dalla memoria di un calcolatore sono codificate utilizzando il codice di Hamming. Si consideri la stringa di 12 bit 101001101110 (il bit meno significativo è a sinistra), risultata della codifica di una parola di N bit secondo il codice di Hamming. **Spiegando bene ogni passo del ragionamento:**

- (1 punto) calcolare N, supponendo di aver fatto uso del numero minimo di bit di controllo necessario per una stringa di 12 bit;
- (VO: 1 punto – NO: 2 punti) scrivere la parola di N bit a partire dalla stringa data;
- (2 punti) indicare eventuali errori nella stringa codificata, specificando quale dei bit è stato alterato.
- (2 punti) Spiegare in maniera chiara e sintetica l'idea alla base del funzionamento del codice di Hamming mediante i diagrammi di Venn.

ESERCIZIO 5 (solo VO: 6 punti)

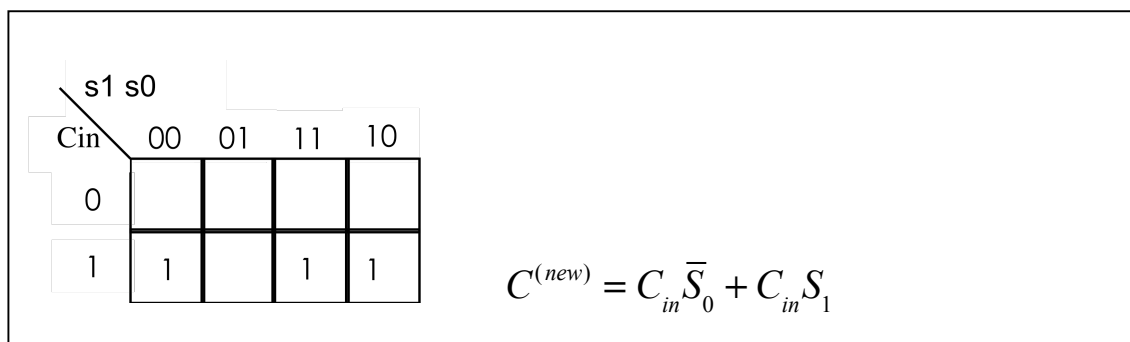
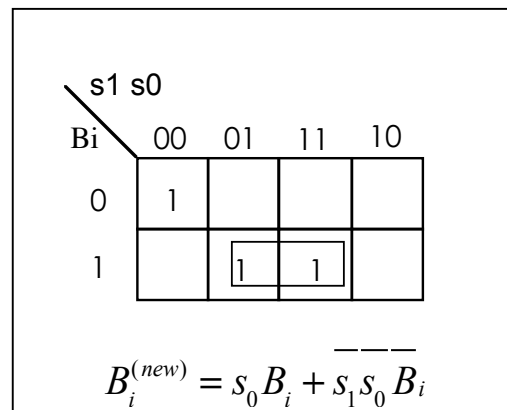
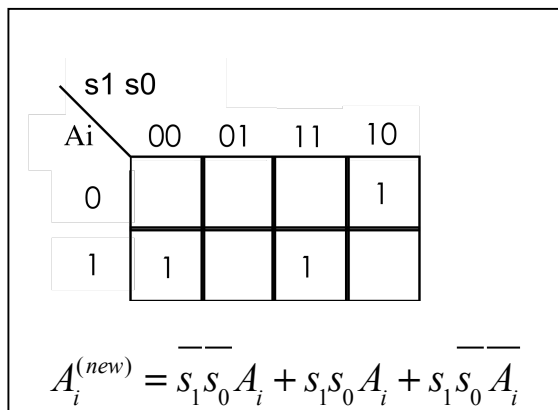
Spiegare in modo chiaro e sintetico la classificazione "strutturale" delle architetture parallele a seconda che si faccia riferimento:

- (3 punti) all'organizzazione della memoria;
- (3 punti) alla rete di interconnessione.

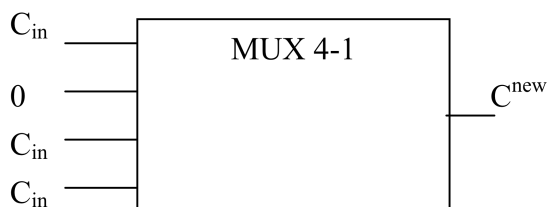
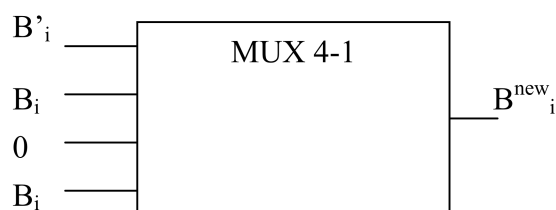
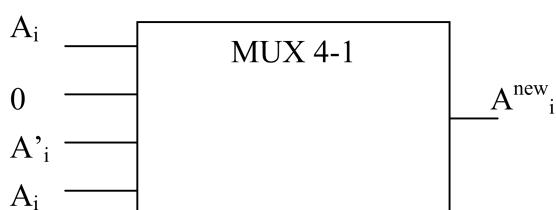
ESERCIZIO 1

Soluzione.

- 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.



- 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

Soluzione:

1. Per indirizzare 256 Mbyte occorre un indirizzo di almeno 28 bit. Per indirizzare il singolo byte all'interno di un blocco occorrono 4 bit ($16 = 2^4$), che coincidono con i 4 bit meno significativi dell'indirizzo di memoria primaria. I restanti 24 bit costituiscono l'indirizzo del "block frame". Per indirizzare la cache, il "block frame" viene interpretato diversamente a seconda che l'indirizzamento sia di tipo "diretto" o "associativo su insiemi".

- Indirizzamento diretto. In questo caso devo poter indirizzare ciascuno dei 32K blocchi contenuti nella cache ($512\text{Kbyte}/(16\text{byte}/\text{blocco})$). Occorrono 15 bit che coincidono con i 15 bit meno significativi del "block frame". Pertanto i 28 bit di indirizzo della memoria primaria vengono interpretati come:

tag	cache index	Offset
9 bit	15 bit	4 bit

- Indirizzamento "associativo su insiemi". In questo caso devo poter indirizzare ciascuno dei 16 insiemi in cui sono suddivisi i blocchi contenuti nella cache ($\frac{512\text{Kbyte}}{\frac{2\text{blocchi}}{\text{insieme}} \cdot \frac{16\text{byte}}{\text{blocco}}} = 16\text{K insiemi}$).

Occorrono 14 bit che coincidono con i 14 bit meno significativi del "block frame"

Pertanto i 28 bit di indirizzo della memoria primaria vengono interpretati come:

tag	cache index	offset
10 bit	14 bit	4 bit

- La memoria è divisa in blocchi di 16 byte ciascuno in modo che la richiesta di un dato non presente in cache causa il trasferimento del blocco a cui appartiene il dato richiesto dalla memoria principale alla cache. Nel caso proposto i dati richiesti sono così suddivisi (N.B. per semplicità i valori di index sono riportati in formato decimale, mentre gli indirizzi in esadecimale):

indirizzi 0000000 ÷ 000000F ⇒ index 0; indirizzi 0000010 ÷ 000001F ⇒ index 1; ... ; 00007F0 ÷ 00007FF ⇒ index 127; 0180000 a 018000F ⇒ index 0; ... ; 01807F0 a 01807FF ⇒ index 127. (N.B. in questa sequenza l'index è identico nel caso di indirizzamento diretto e associativo su insiemi).

La prima volta che viene richiesta la parola 0000000 avremo un "cache miss" che provoca il caricamento del blocco 0 nella linea di cache di indirizzo 0. Le successive richieste dei dati di indirizzo 0000001 ÷ 000000F vengono quindi soddisfatte dalla cache ("cache hit"). Analogamente avviene per tutti i blocchi fino al blocco 127, che vengono allocati in linee consecutive della cache fino alla 127. Quando viene richiesta la parola 0180000 (index 0), nel caso di indirizzamento diretto questa sovrascrive il blocco 0000000 ÷ 000000F e così via per tutte le richieste consecutive. Nel caso di indirizzamento associativo su insiemi a due vie uno stesso index corrisponde a un insieme di due blocchi. Pertanto le parole da 0180000 a 01807FF non sovrascrivono quelle precedentemente inserite, ma vengono memorizzate nel secondo blocco disponibile in ciascun insieme memorizzato.

Nei cicli successivi al primo il processore richiede nuovamente tutti i dati, a partire da 0. Nel caso di indirizzamento diretto avremo un miss per il caricamento della prima parola di ciascun blocco e 15 hit per le parole dello stesso blocco, per tutti i 256 blocchi. Nel caso di indirizzamento associativo su insiemi si hanno solo hit perché tutti i blocchi sono presenti in cache.

In sintesi:

- nel caso di indirizzamento diretto avremo per ciascun ciclo 256 miss, quindi 1280 miss in totale ($256 \cdot 5$);
- nel caso di indirizzamento associativo su insiemi a due vie avremo soltanto 256 miss in totale, tutti relativi al primo ciclo.

Poiché il numero totale di accessi è sempre $256 \cdot 5 \cdot 16$, si ricava:

$H(\text{diretto}) = 0,9375$;

$H(\text{associativo su insiemi}) = 0,9875$.

ESERCIZIO 3 NO**Soluzione.**

```

copy_idx:
    addi $29, $29, -32
    sw $4, 0($29)
    sw $5, 4($29)
    sw $6, 8($29)
    sw $7, 12($29)
    sw $8, 16($29)
    sw $9, 20($29)
    sw $10, 24($29)
    sw $11, 28($29)
    move $8, $0
for:
    beq $8, $7, exit_copy_idx
    lw $9, 0($5)          #carica l'indice in $9
    muli $9, $9, 4        #indice per 4
    lw $11, 0($4)         #$11 <-x[i]
    add $10, $6, $9       #calcola &y[idx[i]]
    sw $11, 0($10)
    addi $4, $4, 4
    addi $5, $5, 4
    addi $8, $8, 1
    j for
exit_copy_idx:
    lw $4, 0($29)
    lw $5, 4($29)
    lw $6, 8($29)
    lw $7, 12($29)
    lw $8, 16($29)
    lw $9, 20($29)
    lw $10, 24($29)
    lw $11, 28($29)
    addi $29, $29, 32
    jr $31

```

ESERCIZIO 3 VO**Soluzione.**

A zero indirizzi	A un indirizzo	A due indirizzi
PUSH D	LOAD B	MOV C,Z
PUSH C	SUB C	SUB B,Z
PUSH B	STORE Z	DIV A,Z
SUB	LOAD A	ADD A,Z
PUSH A	DIV Z	
DIV	ADD D	
ADD	STORE Z	
POP Z		

3. Vedi dispense del corso.

ESERCIZIO 4

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 $N + K = 12$, 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
1	0	1	0	0	1	1	0	1	1	1	0

Dove $c_0 \dots c_3$ sono i quattro bit costituenti il vettore di controllo, e $b_0 \dots b_7$ gli otto bit trasmessi. La sequenza ricevuta è 10111110.

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

$$c'_0 = b_0 \oplus b_1 \oplus b_3 \oplus b_4 \oplus b_6 = 0$$

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

$$c'_2 = b_1 \oplus b_2 \oplus b_3 \oplus b_7 = 0$$

$$c'_3 = b_4 \oplus b_5 \oplus b_6 \oplus b_7 = 1$$

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):

$$e_0 = c_0 \oplus c'_0 = 1$$

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

$$e_2 = c_2 \oplus c'_2 = 0$$

$$e_3 = c_3 \oplus c'_3 = 1$$

Poiché il vettore risultante 1011 non è nullo, vi è un errore nella stringa di 12 bit data e precisamente nella posizione indicata dal vettore di errore tradotto in notazione decimale. Il bit sbagliato è quindi l'undicesimo (b_6), e la parola corretta è 10111100.

4. Si veda il lucido 115 del cap. 4 (memoria).

ESERCIZIO 5

Soluzione.

Vedere le dispense del corso.