

**SOLUZIONI DELLA PROVA SCRITTA DEL CORSO DI
CALCOLATORI ELETTRONICI
NUOVO E VECCHIO ORDINAMENTO DIDATTICO**
18 Luglio 2006

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

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:

s_1	s_0	$c = 0$	$c = 1$
0	0	$A-B+(1\dots 1)$	$A-B$
0	1	B	$B+1$
1	0	A'	$A'+1$
1	1	$A+B$	$A+B+1$

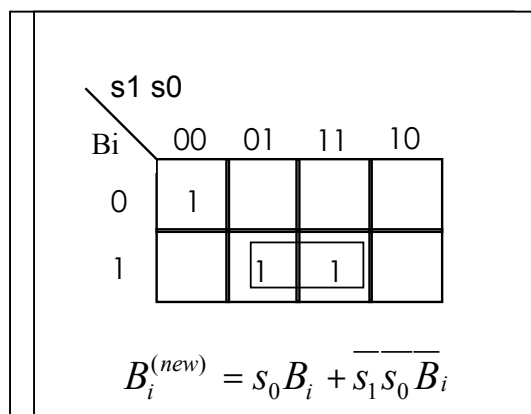
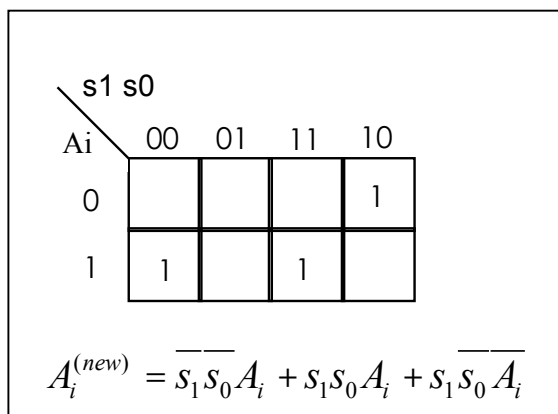
dove s_1 ed s_0 sono due variabili di controllo. Il termine $(1\dots 1)$ indica una stringa di n bit tutti pari ad 1.

- (NO: 4 punti- VO: 3 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.
- (3 punti) Si sostituisca la rete logica al passo precedente con due MUX 4-1.

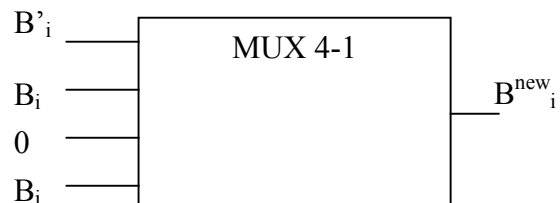
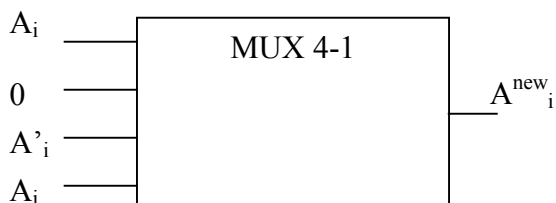
Soluzione.

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 (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	$\text{ACC} / M[X] \rightarrow \text{ACC}$	MUL X1,X2	$M[X1] * M[X2] \rightarrow M[X2]$
MUL	$\text{pop} * \text{pop} \rightarrow \text{push}$	MUL X	$\text{ACC} * M[X] \rightarrow \text{ACC}$		

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

- (5 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 un registro P dove depositare i risultati parziali)

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

Soluzione.

- Ecco tre possibili sequenze delle istruzioni che realizzano quanto richiesto:

A pila	A un indirizzo	A due indirizzi
PUSH A	LOAD B	MOV B,P
PUSH B	ADD C	ADD C,P
PUSH C	MUL A	MUL A,P
ADD	DIV D	MOV D,Z
MUL	STORE Z	DIV P,Z
PUSH D		
DIV		
POP Z		

- Vedere le dispense del corso.

ESERCIZIO 2(solo NO: 8 punti)

Implementare in Assembler MIPS il seguente codice C:

```
int seleziona(int k)
{
    if (k>=0)
        return fattoriale(k);
    else
        return potenza(k,2);
}
```

k è memorizzato in \$4, mentre il valore da restituire deve essere memorizzato in \$5.

Si supponga che la funzione `fattoriale(k)`, ricevendo k in \$4, restituisca k! in \$5 e che la funzione `potenza(k,h)`, ricevendo k in \$4 e h in \$5, restituisca k^h in \$6.

N.B. Non è richiesta l'implementazione delle funzioni `fattoriale` e `potenza`.

Soluzione.

```
seleziona:      addi $29, $29, -8
                sw $8, 0($29)
                sw $31, 4($29)
                slt $8, $4, $0
                bne $8, $0, esegui_potenza
                jal fattoriale
                j fine
esegui_potenza: addi $5, $0, 2
                jal potenza
                move $5, $6
fine:           lw $8, 0($29)
                lw $31, 4($29)
                addi $29, $29, 8
                jr $31
```

ESERCIZIO 3 (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.

- 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 venga usata la modalità di indirizzamento
 - (a) Diretto
 - (b) "associativo su insiemi", e ciascun insieme contenga due blocchi
- 2) (VO: 6 punti - NO: 8 punti) Si consideri la cache di cui alla domanda precedente. Ipotesi 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.

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".
 - (a) 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

- (b) 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

- 2) 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 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 facilmente l'hit ratio H dato che:

$$H = 1 - \text{numeroMiss} / \text{numeroAccessi}$$

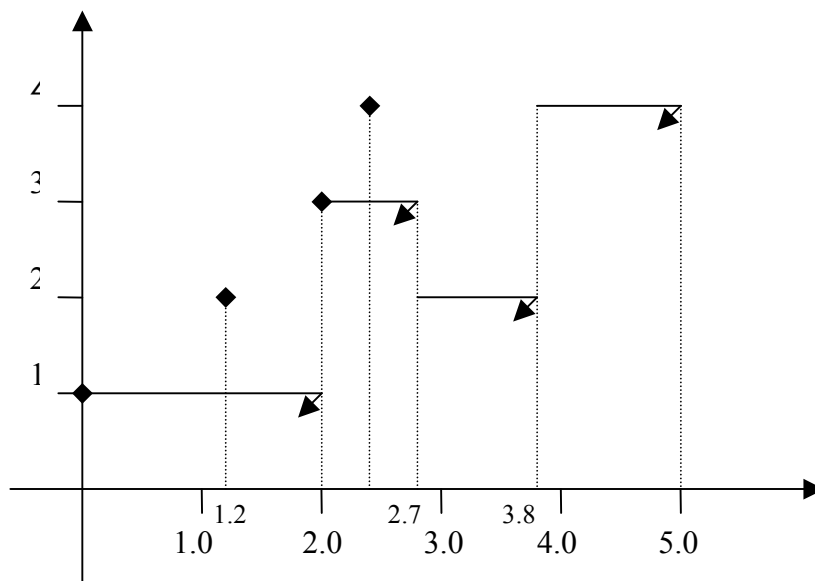
Quindi $H(\text{diretto}) = 0,9375$; $H(\text{associativo su insiemi}) = 0,9875$.

ESERCIZIO 4 (NO: 8 punti - VO: 7 punti)

Sia data la seguente lista di processi (si supponga che l'istante iniziale sia 0):

Job	Tempo di Arrivo	Tempo di CPU richiesto
1	0.0	2.0
2	1.2	1.1
3	2.0	0.7
4	2.5	1.2

Mostrare la sequenza di esecuzione dei job usando un grafico (tempo,job). Calcolare il tempo di *turnaround* medio e il tempo di *turnaround* *pesato* medio, qualora si impieghi la politica di scheduling SJF.



Job	Arrivo	Start	Finish	CPU time	Turnaround	W.turnaround
1	0.00	0.00	2.00	2.00	2.00	1.00
2	1.20	2.70	3.80	1.10	2.60	2.36
3	2.00	2.00	2.70	0.70	0.70	1.00
4	2.50	3.80	5.00	1.20	2.50	2.08
Average					1.95	1.61

ESERCIZIO 5 (solo VO: 5 punti)

Spiegare in modo chiaro e sintetico la gestione della memoria in un sistema operativo, con particolare riferimento alla paginazione, alla segmentazione e ai vantaggi e svantaggi di queste due tecniche.

Soluzione.

Vedere le dispense del corso.